POLITECHNIKA ŚLĄSKA
Biblioteka Główna
w Gliwicach
P. 18177/88

6
1988

informatyka

Cyfrowe generowanie obrazów
Sieć D-Link
Transputer

WYDAWNICTWO NOT SIGMA

Cena 200 zł

informatyka

6 1988

Nr 6
Miesięcznik Rok XXIII
Czerwiec 1988

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

KOLEGIUM REDAKCYJNE:
Mgr Jarosław DEMINET,
dr inż. Wacław ISZKOWSKI,
mgr Teresa JABŁOŃSKA
(sekretarz redakcji),
Władysław KLEPACZ
(redaktor naczelny),
dr inż. Marek MACHURA,
dr inż. Wiktor RZECZKOWSKI,
mgr inż. Jan RYŻKO,
mgr Hanna WŁODARSKA,
dr inż. Janusz ZALEWSKI
(zastępca redaktora naczelnego).

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

RSW „PRASA-KSIĄŻKA-RUCH"
PRASOWE ZAKŁADY GRAFICZNE
ul. Dworcowa 13, 85-950 BYDGOSZCZ
Zam. 2370/88. U-40
Obj. 4,0 ark. druk. Nakład 8950 egz.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 200 zł
Prenumerata roczna 2400 zł

WYDAWNICTWO NAUKOWO-TECHNICZNE
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Cyfrowe generowanie obrazów (1)
Jan Zabrodzki 1

Języki wysokiego poziomu do przetwarzania napisów – COMIT, SNOBOL4 i Icon (3)
Ralph E. Griswold, Madge T. Griswold 4

Lokalna sieć komputerowa D-Link (1)
Andrzej Turowicz 7

Transputer – narzędzie przetwarzania równoległego
Dariusz Caban 10

Sieć komputerowa Jednolitego Systemu SKJS2, wersja 1
Andrzej Huzar, Andrzej Kaliś 14

PROMIS – System Zarządzania Bazą Danych (2)
Halina Ciechomska, Teresa Wójciekian, Ewa Zaborowska 16

Turbo C. Preprocesor i makrodefinicje
Jan Bielecki 19

Jeszcze raz o Turbo Pascalu
opr. *Mariusz Kuc* 21

ZE ŚWIATA 26

W pogoni za efektywnością komputerów
Kompilator Ady dla IBM PC/AT
Rodzina komputerów ETA10
Bardzo szybka jednostka centralna firmy Inmos
Zagrożenie planów opracowania szybkich komputerów w Japonii

TERMINOLOGIA 31

System plików (1)

W NAJBLIŻSZYCH NUMERACH:

- **Hans-Jurgen Appelrath charakteryzuje system wyszukiwania informacji ODIR z wykorzystaniem laserowej płyty wizyjnej**
- **Sławomir Błaszczak omawia język Occam przeznaczony do programowania transputerów**
- **Piotr Fuglewicz i Tadeusz Korniak przedstawiają założenia i sposób realizacji wielodostępnego systemu operacyjnego MASTER**
- **Wojciech Mokrzycki prezentuje techniki obrazowania informacji o trójwymiarowych zdyskretyzowanych scenach**
- **Witold Abramowicz opisuje nowy sposób pamiętania w systemach informacyjnych**
- **Mariusz Kuc charakteryzuje skrośny system firmy Alsys wspomagania projektowania w Adzie**

JAN ZABRODZKI
Instytut Informatyki
Politechnika Warszawska

POLSKIE TOWARZYSTWO INFORMATYCZNE

Pierwsza wersja artykułu została przedstawiona podczas Szkoły Jesiennej PTI „Współczesne kierunki rozwoju informatyki" – Mrągowo, 7–11 grudnia 1987 r.

POLITECHNIKA ŚLĄSKA Biblioteka Główna w Gliwicach P. 1817/88

Cyfrowe generowanie obrazów (1)

Rozwój technologii układów scalonych pociąga za sobą rozwój wielu innych dziedzin. Dotyczy to m. in. rastrowej techniki graficznej. Współczesne układy VLSI oraz monitory graficzne umożliwiają konstruowanie stosunkowo tanich urządzeń dla cyfrowego generowania i wyświetlania obrazów o dobrej jakości i w stosunkowo krótkim czasie. Obserwuje się rozwój konstrukcji takich urządzeń jak stacje graficzne, zdolnych do samodzielnej pracy bądź do współpracy z większym systemem, celem odciążania go od obliczeń typowo „graficznych".

Celem artykułu jest przedstawienie podstawowych problemów związanych z cyfrowym generowaniem obrazów przy wykorzystaniu techniki rastrowej, zwłaszcza z punktu widzenia konstrukcji sprzętu.

Po wprowadzeniu w zagadnienia techniki rastrowej przedstawione są kolejno problemy związane z: konstruowaniem pamięci obrazu, wyświetlaniem na monitorach kolorowych (łącznie z omówieniem ich podstawowych cech) i generowaniem realistycznych obrazów w czasie rzeczywistym. Pierwszą część artykułu kończy prezentacja najnowszych układów VLSI, projektowanych dla potrzeb konstruowania urządzeń grafiki komputerowej.

W drugiej części przedstawiono współczesne konstrukcje stacji graficznych oraz wybraną metodę rozwiązywania zagadnień związanych z wyznaczaniem obiektów widocznych i ich oświetleniem – tzw. metodę śledzenia promieni. Metoda ta rokuje duże nadzieje, jeśli chodzi o możliwość cyfrowego generowania realistycznych obrazów, jednak jej złożoność obliczeniowa przekracza możliwości obecnie produkowanych specjalizowanych urządzeń i jej stosowanie wymaga dużych systemów komputerowych. Przykłady takich systemów są przedstawione w zakończeniu.

TECHNIKA RASTROWA

W technice rastrowej obraz jest tworzony w postaci prostokątnego zbioru punktów. Każdemu z tych punktów jest przypisany odpowiedni odcień szarości lub kolor. Efekt spójności obrazu uzyskuje się dzięki skończonej rozdzielczości oka ludzkiego. Jakość obrazu zależy od liczby punktów i jest tym większa im większa jest rozdzielczość obrazu określona przez wymiary tablicy punktów.

Cyfrowe generowania obrazów wiąże się z koniecznością rozwiązywania wielu problemów, zwłaszcza wtedy gdy celem jest generacja obrazów jak najwierniej odzwierciedlających obiekty rzeczywiste. Problemy te wynikają głównie z konieczności reprezentowania obrazu w postaci skwantowanej i za pomocą sprzętu o określonych wartościach parametrów funkcjonalnych. Rozwój techniki rastrowej jest związany więc z jednej strony z poszukiwaniem coraz to doskonalszych algorytmów dla generowania obrazów, a z drugiej strony ze stałym ulepszaniem parametrów sprzętu obliczeniowego i wyświetlającego generowane obrazy. Obie te grupy zagadnień wzajemnie się przenikają i uzupełniają. Opracowywane są algorytmy, które pozwalają w sposób optymalny wykorzystać istniejący sprzęt, jak też stosowane są rozwiązania sprzętowe optymalizowane z punktu widzenia realizacji określonych algorytmów.

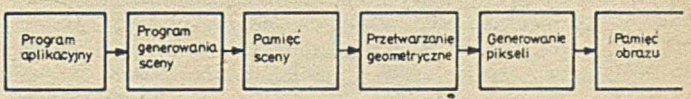
W największym uproszczeniu każdy system grafiki rastrowej można przedstawić za pomocą schematu blokowego jak na rys. 1.

Centralnym elementem systemu jest pamięć obrazu, w której zapisana jest informacja o każdym punkcie (pikselu) obrazu. Informacje te są wytwarzane przez blok cyfrowego generowania obrazu. Natomiast za prezentację obrazu zawartego w pamięci obrazu odpowiedzialny jest blok wyświetlania.

Rys. 1. Schemat blokowy systemu rastrowego

Pojemność pamięci obrazu musi być co najmniej równa liczbie pikseli wyświetlanego obrazu. Przykładowo, dla ekranu o rozdzielczości 1024 × 1024 i ośmiobitowej informacji o kolorze piksela minimalna pojemność pamięci obrazu wynosi 1 MB. O ile uzyskanie pamięci o odpowiedniej pojemności nie stanowi obecnie istotnego problemu, to trudności związane są z zapewnieniem dostępu do tej pamięci przez oba współpracujące z nią bloki.

Dla uniknięcia efektu migotania obraz powinien być wyświetlany z częstotliwością rzędu 60 Hz. Oznacza to w krańcowym wypadku konieczność zapisywania do pamięci 60 obrazów w ciągu sekundy, czyli generowania informacji o kolejnych pikselach (co wiąże się z wykonaniem pewnej liczby operacji arytmetycznych) i zapisywaniu ich do pamięci co około 16 ns[1]. Równocześnie z tą samą częstotliwością, 60 obrazów na sekundę, muszą być wyprowadzane informacje na ekran. Uwzględniając ograniczenia wynikające z konstrukcji monitorów okazuje się, że informacja o kolejnych pikselach powinna być odczytywana z pamięci co około 11 ns. Biorąc łącznie oba aspekty pod uwagę, czas dostępu do pamięci powinien być na poziomie 5 ns. Jak dotychczas nie istnieją pamięci o takich czasach dostępu i wymaganej pojemności.



Rys. 2. Zadania realizowane przez blok cyfrowego generowania obrazu

Na rys. 2 przedstawiono zestaw zadań realizowanych przez blok cyfrowego generowania obrazu. Informacje wyjściowe dla procesu generowania obrazu są dostarczane przez program aplikacyjny. Pozwalają one określić poszczególne obiekty sceny a następnie skomponować scenę z tych obiektów. Opis sceny, np. w postaci zbioru wielokątów z odpowiednimi atrybutami, jest pamiętany w pamięci sceny. Opis trójwymiarowej sceny służy z kolei za podstawę dla utworzenia obrazu sceny w dwuwymiarowej płaszczyźnie ekranu. Dokonują tego odpowiednie procedury geometryczne (zmiana układu współrzędnych, obrotu, przesunięcia, skalowanie, rzuty perspektywiczne, obcinanie itd.). Często, dla uproszczenia obliczeń, operacji tych dokonuje się na ograniczonym zbiorze punktów, np. na zbiorze wierzchoł-

[1] Dla porównania przypomnijmy, że czas wykonania pojedynczej instrukcji przez mikroprocesor 8086 wynosi kilkaset ns.

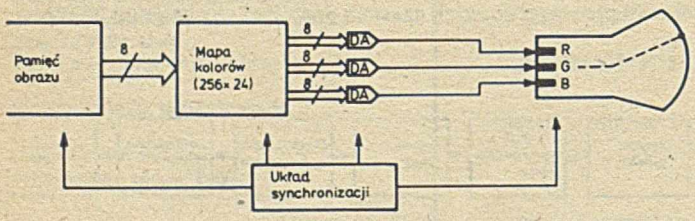
ków wielokątów tworzących scenę. W takiej sytuacji po zakończeniu obliczeń konieczne jest generowanie informacji o pozostałych punktach obrazu, np. przez wypełnianie wielokątów odpowiednimi kolorami. Zależnie od stopnia złożoności sceny i wymaganej wierności odtworzenia sceny rzeczywistej wykonuje się różne zestawy algorytmów o różnej złożoności obliczeniowej.

Dla zilustrowania problemu rozważmy operacje związane tylko z transformacjami geometrycznymi. W celu wykonania operacji skalowania, przesunięcia, obrotu i rzutu perspektywicznego jednego punktu o trzech wymiarach trzeba wykonać 20 operacji zmiennoprzecinkowych (9 dodawań, 9 mnożeń i 2 dzielenia). W tabeli podane są przykładowe czasy transformacji obrazu o 2000 wierzchołków za pomocą programu w systemie z mikroprocesorem 8086, przy wykorzystaniu koprocesora arytmetycznego 8087 i specjalnych układów mnożących (np. Am29325 lub Weitek 1032) [3].

Czasy transformacji obrazu o 2000 wierzchołków

Metoda	Czas mnożenia [μs]	Czas dodawania [μs]	Czas dzielenia [μs]	Czas transformacji 2000 wierzchołków	Liczba obrazów na sekundę
Programowa (8086)	1600	1600	3200	70,4 s	0,015
Z koprocesorem (8087)	19	17	39	0,804 s	1,25
Układ mnożący	0,2	0,2	0,8	10,4 ns	96

Blok wyświetlania obrazu zawiera układy, które umożliwiają przesłanie z odpowiednią szybkością informacji pobieranych z pamięci obrazu do monitora. Układy te muszą zapewnić konwersję sygnałów cyfrowych na postać analogową, wymaganą przez układy wejściowe monitora, oraz wytworzenie sygnałów synchronizujących proces wyświetlania. Ponadto często stosowanym elementem w bloku wyświetlania jest tablica kolorów. Jest to pomocnicza pamięć, na której wejścia adresowe jest podawana z pamięci obrazu informacja o kolorze piksela, np. 8-bitowa, i na której trzech wyjściach, 4- lub 8-bitowych, pojawiają się kody składowych koloru R, G, B. Pozwala to na korzystanie z dużej gamy kolorów za pomocą krótkich słów niosących informację o kolorze piksela.



Rys. 3. Blok wyświetlania z mapą kolorów

Dla przykładu, pamięć kolorów o 256 słowach 24-bitowych pozwala korzystać z palety 16 milionów kolorów. Pamięć kolorów musi być pamięcią bardzo szybką. W czasie przeznaczonym na wyświetlenie jednego piksela (w podanym wyżej przykładzie 11 ns) konieczne jest dokonanie zarówno konwersji kolorów jak i konwersji cyfrowo-analogowej. Rys. 3 poglądowo przedstawia omówione wyżej funkcje bloku wyświetlania.

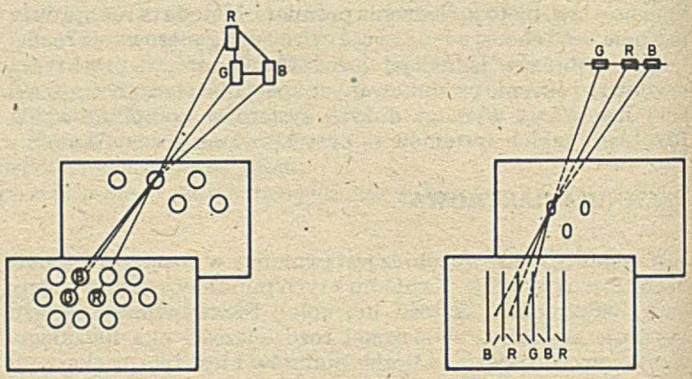
MONITORY KOLOROWE

Podstawowym elementem monitora kolorowego jest kineskop kolorowy. Pozostałe układy monitora służą do sterowania kineskopem. Monitory kolorowe wysokiej jakości przyjmują informację przeznaczoną do wyświetlania w postaci napięć, podanych na wejścia sterujące działami elektronowymi związanymi z kolorami czerwonym (R), zielonym (G) i niebieskim (B).

Przy odtwarzaniu obrazu kolorowego wykorzystuje się właściwości oka ludzkiego i jego zdolność widzenia barw. Oko ludzkie nie rozróżnia poszczególnych elementów kolorowych, jeżeli powierzchnie elementów barwnych oglądane z pewnej odległości są odpowiednio małe. Ekran kineskopu kolorowego powinien świecić w trzech kolorach podstawowych R, G, B. Są to kolory, które poprzez odpowiednie sumowanie pozwalają uzyskać inne kolory. Dla przykładu, kolor biały uzyskuje się przy sumowaniu w proporcjach 0,3R, 0,59G, 0,11B. Bezwładność oka w widzeniu barw umożliwia odtwarzanie obrazu kolorowego zarówno przy jednoczesnym, jak i przy kolejnym rozświetlaniu luminoforów o podstawowych kolorach R, G, B. Przy zasadzie kolejnego świecenia musi być zachowany warunek dostatecznie dużej częstotliwości rozświetlania luminoforów.

Obecnie w monitorach kolorowych są stosowane trzy typy kineskopów maskowych, tzw. kineskopy delta, PIL (ang. precision in line) oraz trynitron.

W kineskopach typu delta luminofory są nałożone na ekran w postaci okrągłych „pastylek" o średnicy rzędu ułamka milimetra. Pastylki te są ułożone wzdłuż linii poziomych w kolejności R, G, B, R, G, B, R itd. W sąsiednich liniach pastylki są przesunięte względem siebie tak, żeby trzy sąsiednie pastylki wyznaczały wierzchołki trójkąta równobocznego. Liczba takich trójek (triad) zależy od rozmiarów ekranu. W kineskopie znajdują się trzy wyrzutnie elektronów, każda przeznaczona do pobudzania pastylek innego koloru, rozmieszczone w wierzchołkach trójkąta równobocznego. W odległości kilkunastu milimetrów przed ekranem znajduje się cienka blacha z dużą liczbą otworów, tzw. maskownica. W celu uzyskania obrazu o dobrej jakości, trzy strumienie elektronowe powinny przecinać się w środku każdego otworu maskownicy, a następnie „rozchodzić się" padając na odpowiednie pastylki luminoforów. Dla danego strumienia, przy dowolnym jego odchyleniu, pastylki dwóch innych luminoforów powinny być zasłonięte przez maskownicę. Przedstawioną zasadę poglądowo ilustruje rys. 4a. Strumienie elektronów z trzech wyrzutni R, G, B, są odchylane w kierunkach pionowym i poziomym za pomocą jednego wspólnego zespołu odchylającego.



Rys. 4. Zasada pracy kineskopu typu delta (a) i typu PIL (b)

W kineskopach typu PIL ekran jest pokryty wąskimi paskami luminoforów ułożonymi pionowo. Otwory w maskownicy mają kształt pionowych szczelin o jednakowej długości. Działa elektronowe są umieszczone obok siebie w płaszczyźnie poziomej. Zasadę działania kineskopu ilustruje rys. 4b. W porównaniu z metodą delta, w technice PIL upraszcza się problem zbieżności strumieni elektronowych oraz upraszczają się w związku z tym układy sterowania kineskopem. Zaletą metody delta jest większa rozdzielczość.

Kineskopy typu trynitron są stosowane w zasadzie wyłącznie w monitorach firmy Sony. Podobnie jak w kineskopach typu PIL, luminofor jest naniesiony w postaci wąskich pionowych pasków i na maskownicy znajdują się pionowe szczeliny. Zasadnicza różnica polega na odmiennej konstrukcji działa. W systemie tym stosowane jest jedno działo o trzech katodach umiesz-

czonych poziomo. Rozwiązanie to pozwala uzyskać bardzo dobrą rozdzielczość, jest jednak trudne technologicznie.

Rozwój monitorów kolorowych do zastosowań w technice graficznej zmierza w kierunku uzyskania jak największej rozdzielczości. Wśród monitorów produkowanych obecnie i dostępnych handlowo największa rozdzielczość wynosi 1280 × 1024 (przy odległości między triadami – 0,31 mm). Są to monitory 19-calowe o częstotliwości przeszukiwania poziomego 64 kHz i paśmie wizyjnym rzędu 100 MHz.

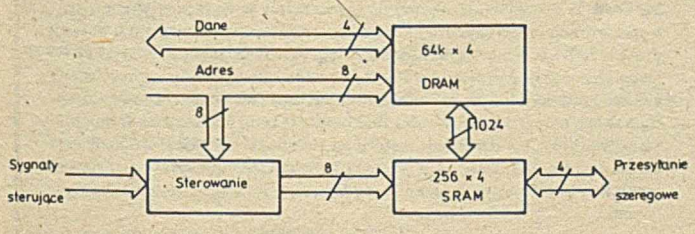
Sterowanie ruchem wiązek elektronów w kineskopie jest następujące. Poczynając od górnego lewego rogu, strumień przesuwa się poziomo w prawo, wybierając wszystkie punkty należące do pierwszej linii obrazu. Z kolei, następuje przesunięcie strumienia do początku drugiej linii i rozpoczyna się wybieranie punktów należących do tej linii itd. W czasie każdego powrotu strumienia do początku linii nie jest wyświetlana żadna informacja (promień jest wygaszany). Z tego punktu widzenia czas powrotu jest czasem martwym. Podobnie czasem martwym jest czas powrotu plamki z punktu końcowego ostatniej linii do początku ekranu (przy wyświetlaniu z tzw. wybieraniem międzyliniowym w pierwszej fazie wybierane są linie parzyste, a w drugiej linie nieparzyste).

Dla monitora o rozdzielczości 1208 × 1024 przy odświeżaniu obrazu 60 razy na sekundę przyjmując, że czas powrotu pionowego plamki jest około 600 μs otrzymuje się, że częstotliwość wybierania linii poziomych jest około 64 kHz. Stąd czas na wyświetlenie jednej linii wynosi około 13 μs (po odjęciu czasu powrotu poziomej plamki 4 μs). Przy 1280 punktach wyświetlanych w linii oznacza to, iż czas wyświetlania punktu jest rzędu 10 ns. Stąd wymagane pasmo wizyjne jest rzędu 100 MHz. Podane parametry ma np. monitor HM 4619 firmy Hitachi.

UKŁADY VLSI DLA ZASTOSOWAŃ GRAFICZNYCH

W ostatnich kilku latach opracowano wiele specjalizowanych układów scalonych przeznaczonych do konstruowania urządzeń grafiki komputerowej. Dzięki tym układom stało się możliwe opracowanie urządzeń o szybkości i jakości tworzenia obrazów porównywalnych z parametrami osiąganymi do niedawna jedynie za pomocą dużych systemów, budowanych z wykorzystaniem układów segmentowych i techniki ECL. Wśród nowych układów wyróżnić można kilka grup.

W grupie pamięci, obok różnego typu układów pamięci uniwersalnych DRAM o pojemności 1 Mb, pojawiły się układy pamięci VRAM (ang. video RAM). Są to pamięci o dwóch rodzajach odstępu: klasycznym – równoległym i nowym – szeregowym. Schemat blokowy wyjaśniający konstrukcję tego typu pamięci jest podany na rys. 5 (jest to schemat układu Am 90C644 firmy AMD). Obok bloku pamięci dynamicznej o pojemności 64 Kb × 4 (i organizacji wewnętrznej 256 × 256 × 4) w strukturze znajduje się rejestr szeregowy o pojemności 256 słów 4-bitowych. Poza normalnym trybem zapisu i odczytu pamięci (przy czasie dostępu 190 ns) możliwe jest równoczesne przepisanie w czasie jednego cyklu dostępu do pamięci 256 słów 4-bitowych do rejestru szeregowego. Od tej chwili informacja może być wyprowadzana z rejestru szeregowo, niezależnie od pracy pamięci w trybie równoległego dostępu (z dużą częstotliwością rzędu 25 MHz). Podobnie, w razie potrzeby, można wprowadzać informację szeregowo, a następnie w czasie jednego cyklu zapisać ją do matrycy pamięci DRAM. Zasadniczą zaletą tego typu pamięci jest wzrost procentowego udziału czasu dostępu do pamięci dla celów wprowadzania nowej informacji. Zaleta ta jest mniej istotna w wypadku dublowania pamięci obrazu. Przy takiej konstrukcji, do jednego bloku pamięci jest wprowadzany nowy obraz, podczas gdy z drugiego bloku obraz jest wyprowadzany w celu wyświetlania. Po zakończeniu pełnego cyklu role bloków pamięci zmieniają się.



Rys. 5. Schemat blokowy pamięci z dostępem równoległym i szeregowym

Dla ułatwienia konstrukcji bloków wyświetlania opracowano różnego rodzaju zestawy układów przetworników wizyjnych cyfrowo-analogowych. Wiele z nich oprócz przetworników zawiera inne pomocnicze układy, np. rejestry czy tablicę kolorów. Dostępne są układy z pojedynczymi przetwornikami, jak też z zestawami trzech przetworników DA 4- lub 8-bitowych i częstotliwościach konwersji do 200 MHz (np. TEK 8120 firmy Tektronix lub AH 8308 firmy Analogic – potrójny zestaw, 150 MHz).

Dla celów generacji sygnałów sterujących wyświetlaniem informacji są produkowane specjalne układy, zarówno do zastosowań znakowych, jak i graficznych. Jednym z bardziej popularnych układów tego typu jest układ 6845 firmy Motorola, stosowany m. in. w mikrokomputerach IBM PC. Bardziej zaawansowane układy są przystosowane do pracy w systemach typowo graficznych. Można tu wymienić układy MC 68486/487, 82716, TMS34061 oraz Am 8150. Umożliwiają one sterowanie wyświetlaniem przy rozdzielczości do 640 × 480 pikseli. Niektóre z nich mają wbudowane funkcje zarządzania systemem okien.

Inna grupa układów, tzw. sterowniki ze stałą listą rozkazów graficznych, ma zadanie odciążyć procesor centralny od funkcji związanych z wypełnianiem pamięci obrazu. Jednym z najbardziej rozpowszechnionych układów tego typu jest sterownik graficzny μPD 7220 firmy NEC. Układ ten, oprócz funkcji sterowania wyświetlaniem oraz odświeżaniem pamięci DRAM, realizuje funkcje rysowania linii, łuków, prostokątów, znaków 8 × 8, zapełnianie pól, powiększanie fragmentów obrazu oraz przesuwanie obiektów po ekranie. Najszybsze wersje tego układu pozwalają wyrysować piksel w czasie 500 ns. Podstawową wadą tego układu jest rozłączność w czasie funkcji tworzenia obrazu i wyprowadzania go na ekran. Próbę rozwiązania tego konfliktu podjęli projektanci układu sterownika HD 63484-8 firmy Hitachi. W układzie tym w podstawowym cyklu 1000 ns przewidziano czas zarówno na zapis do pamięci obrazu jak i na odczyt dla celów wyświetlania. Układ wykonuje 38 rozkazów rysowania. Może on sterować pamięcią obrazu o pojemności do 2 MB z 1, 2, 4, 8 lub 16 bitami na piksel.

Do współpracy z pamięciami video RAM przystosowany jest układ sterownika Am 95C60 AMD. Jest to układ o strukturze 16-bitowej, realizujący 60 rozkazów graficznych. Maksymalna rozdzielczość sterowanego układu wynosi 4096 × 4096. Układ ma wbudowany algorytm wygładzania linii z wykorzystaniem dwóch lub czterech poziomów cieniowania.

Najbardziej złożonymi układami projektowanymi z myślą o zastosowaniach graficznych są układy procesorów graficznych. Przedstawicielem tej grupy układów jest procesor TMS 34010 firmy Texas Instruments. Jest to układ 32-bitowego mikroprocesora ze zredukowaną listą rozkazów o szybkości 6 MIPS. Układ jest wykonany w technologii CMOS (o szerokości ścieżek 1,8 μm) i zawiera około 180 000 tranzystorów. Pozwala na wykonywanie operacji przesyłania bloków pikseli. W układzie TMS 34010 wbudowana jest pamięć podręczna (ang. cache) o pojemności 256 bajtów. Umożliwia to pamiętanie 128 rozkazów i przyspieszanie wykonywania pętli.

Obok samych układów scalonych możliwe jest korzystanie z gotowych pakietów grafiki kolorowej, przystosowanych do współpracy z określonymi systemami. Przykładowo, firma Imgraph produkuje pakiet AGC-1024-8 przystosowany do współpracy z systemami IBM PC/XT/AT/RT. W pakiecie tym wykorzystano następujące układy graficzne: sterownik HD–63484, pamięć video RAM oraz układ przewodników wizyjnych DA AMD 8151. Pakiet umożliwia wyświetlanie z częstotliwością 60 Hz obrazów o rozdzielczości 1024 × 1024. Każdy piksel jest reprezentowany za pomocą 8 bitów, co daje możliwość wyświetlania 256 kolorów z palety 16,8 miliona kolorów. Pakiet realizuje podstawowe instrukcje graficzne oraz operacje na blokach bitów, sprzętowe obcinanie, podział ekranu. Maksymalna szybkość wprowadzania informacji do pamięci obrazu wynosi 5 mln pikseli/s, a częstotliwość wyprowadzania na ekran 87 MHz.

LITERATURA

[1] Demetrescu D.: Moving pictures. Byte, pp. 207–217, November 1985.

RALPH E. GRISWOLD
MADGE T. GRISWOLD
Department of Computer Science
Uniwersytet stanu Arizona, Tucson
USA

Języki wysokiego poziomu do przetwarzania napisów – COMIT, SNOBOL4 i Icon (3)

Aczkolwiek SNOBOL4 stanowi silne narzędzie manipulowania napisami i pozwala zaprogramować w sposób łatwy i zwarty wiele skomplikowanych zadań, jest cokolwiek niezgrabny. Przykładowo, brak struktur sterowania zmusza programistę do budowania pętli przy użyciu etykiet, wyrażeń warunkowych i skoków.

SNOBOL4 ma złożoną budowę. Zawiera bowiem konwencjonalne środki do obliczeń numerycznych i innych, nie związanych z przetwarzaniem napisów, natomiast w sferze działań dotyczących dopasowywania wzorców istnieje odrębny zestaw operacji powiązanych ze specyficznym mechanizmem nawrotów w sterowaniu. Rozdział między tymi dwoma aspektami języka SNOBOL4 ilustrują choćby dwie formy przypisania:

s = "Hello world"

w konwencjonalnej składowej języka i

LEN (11) . s

w składowej dotyczącej dopasowywania wzorców. Tym samym podobne działania mają odmienne postaci, a programista musi kształtować konstrukcje programowe w różny sposób, w zależności od kontekstu.

Język programowania Icon[1], opracowany w Uniwersytecie stanu Arizona przez Ralpha E. Griswolda, Davida R. Hansona i Johna T. Korba, wyewoluował z dążeń do przezwyciężenia podobnych problemów (niejednorodności) i chęci odnalezienia nowych sposobów wyrażania wysoko zorganizowanych działań przetwarzania napisów. Icon pod wieloma względami jest podobny do SNOBOLU4. Pojęcie łańcucha jest takie samo, w obu językach podobnie zarządza się strukturami danych. Icon i SNOBOL4 różnią się w obu istotnych aspektach:

- Icon ma konwencjonalne struktury sterowania, a jego składnia nawiązuje do rozwiązań zastosowanych w najnowszych językach programowania;
- konwencjonalne środki obliczeniowe i mechanizmy dopasowywania wzorców są w języku Icon zintegrowane i koncepcyjnie spójne, co umożliwia stosowanie techniki nawrotów w sterowaniu w obliczeniach każdego rodzaju.

Jednocześnie, Icon nie zawiera niektórych spośród bardziej ezoterycznych cech języka SNOBOL4, w tym odwołań pośrednich i kompilacji w toku programu.

Operacje warunkujące i generatory

Icon zachowuje snobolową koncepcję wyrażeń warunkujących, co więcej, używa właściwości spełniania się bądź niespełniania wyrażeń do nadzorowania wykonania rozmaitych konwencjonalnych struktur. Przykładowo:

```
while line := read () do
  write (line)
```

kopiuje plik wejściowy na wyjście. Klauzula **do** nie jest niezbędna, a powyższe wyrażenie można napisać w sposób bardziej zwięzły:

```
while write (read())
```

co da się odczytać następująco: „dopóki udaje się czytać, wypisuj każdy przeczytany wiersz na wyjście".

Podobnie jak w języku SNOBOL4, istnieje wiele wyrażeń określających warunki. Wyrażenia te, gdy się spełniają, wytwarzają użyteczne (w tradycyjnym rozumieniu) wyniki. Za przykład może posłużyć wyrażenie:

find (s1,s2)

które określa miejsce występowania podłańcucha **s1** w łańcuchu **s2**, jeśli podłańcuch taki istnieje, w przeciwnym zaś razie się nie spełnia. Załóżmy, na przykład, że **s2** ma wartość:

"try backtracking"

Wówczas

find ("y",s2)

wytwarza wartość 3, a

[1] Jak wyjaśniają E. R. Griswold i D. R. Hanson w Icon Newsletter (June, 3, 1980), nazwa języka Icon nie jest skrótem żadnych wyrazów. W uzasadnieniu wyboru nazwy Icon dla języka programowania autorzy podają dwa znaczenia tego wyrazu w języku angielskim: znak ikoniczny (czyli znak, którego forma jest podobna do tego, co ten znak oznacza) oraz ikona (przyp. tłum.).

RALPH E. GRISWOLD jest absolwentem Uniwersytetu w Stanford, gdzie w roku 1956 uzyskał dyplom (B.S.) w dziedzinie fizyki a następnie tytuł magistra (M.S.) i doktora (Ph.D.) w dziedzinie inżynierii elektrycznej (electrical engineering) – 1960–1962 r. Od roku 1962 pracownik Bell Laboratories, w roku 1969 mianowany szefem Departamentu Badań Języków Programowania. Działo się to w latach, gdy powstawały odmiany SNOBOLu. W roku 1971 otrzymał nominację profesora informatyki w Uniwersytecie stanu Arizona. Tam pod jego kierunkiem opracowano język Icon. Prócz projektowania i realizacji języków programowania, jego zainteresowania badawcze obejmują programowanie nienumeryczne i systemy opracowywania dokumentów.

MADGE T. GRISWOLD ukończyła uniwersytet w Syracuse w roku 1962 o specjalizacji dziennikarskiej i lingwistycznej. W tym samym roku podjęła pracę w Bell Laboratories jako redaktor techniczny, by w roku 1969 przejść do Departamentu Badań Języków Programowania i kontynuować prace badawcze nad systemami opracowywania dokumentów i komputerowo wspomaganym składaniem tekstów. W roku 1974 uzyskała tytuł magistra nauk humanistycznych (M.A.) w dziedzinie historii na Uniwersytecie stanu Arizona. Jest niezależną pisarką (free-lance writer) i konsultantem do spraw zastosowań techniki obliczeniowej i technik wydawniczych wspomaganych komputerami.

find ("b",s2)

nie spełnia się.
PL/I ma podobną funkcję, o nazwie INDEX. Wszelako tam, zamiast niespełnienia – jeśli podłańcuch nie istnieje – funkcja INDEX oddaje wartość 0. W konsekwencji w PL/I, aby określić czy wartość funkcji INDEX stanowi dopuszczalny numer znaku, czy tylko sygnalizuje niemożność odnalezienia takowego, należy wykonać dodatkowe sprawdzenie wartości INDEX.

W języku SNOBOL4 warunek dopasowania składowej wzorca może się nie spełnić, zostać spełniony, względnie – dopasowanie składowej wzorca może stać się źródłem ciągu wyników, jeśli odbywa się dopasowywanie z nawrotami. W Iconie uogólniono tę koncepcję na wszelkie wyrażenia. Wyrażenie w Iconie może nie wytwarzać żadnego wyniku[2] (niepowodzenie), może wytwarzać jeden wynik, lecz także może być źródłem wielu wyników. Wyrażenie produkujące więcej niż jeden wynik nazywa się **generatorem.**

Wykorzystanie zdolności wytwarzania więcej niż jednego wyniku można zilustrować, kontynuując przykład poprzedni, za pomocą wyrażenia:

find ("t",s2)

które może przyjąć jako swoje wyniki zarówno 1, jak i 9. Użycie podobnego generatora w kontekście wymagającym tylko jednego wyniku powoduje wytworzenie tylko pierwszego wyniku z potencjalnej sekwencji wyników generatora. Toteż w przypisaniu:

i := find("t",s2)

zmienna **i** otrzymuje wartość 1.

Istnieją dwa konteksty, w których generator może wytworzyć więcej niż jeden wynik: **iteracja** i **obliczenie ukierunkowane celem** ang. (goal-directed evaluation). Iteracja, zapisywana w postaci wyrażenia:

every $expr_1$ **do** $expr_2$

powoduje ustawiczne podejmowanie obliczania wyrażenia $\mathbf{expr_1}$, w celu uzyskania ciągu wszystkich możliwych wyników, jakie dane wyrażenie jest w stanie wyprodukować. Dla każdego wyniku wytworzonego przez $\mathbf{expr_1}$ oblicza się $\mathbf{expr_2}$. W konsekwencji:

```
every i := find("t",s2) do
  write (i)
```

spowoduje wypisanie 1 i 9. Człon **do** można pominąć, zapisując to wyrażenie w zwięźlejszej postaci:

every write (find("t",s2))

Obliczanie sterowane celem jest niejawną właściwością algorytmu ewaluacji wyrażeń w Iconie i przypomina postępowanie przy dopasowywaniu wzorców w języku SNOBOL4. W Iconie algorytm obliczania wyrażeń, dążąc zawsze do spełnienia (wyprodukowania wyniku innego niż żaden) całego wyrażenia, powoduje automatyczne podejmowanie działania generatorów dla wytworzenia dodatkowych wyników, jeśli dotychczas uzyskane rezultaty nie prowadzą do tego celu. Na przykład, jedna z warunkujących operacji Iconu:

i > j

spełnia się, jeśli wartość **i** jest większa niż wartość **j** i nie spełnia – w przeciwnym razie. Jeśli argument tego rodzaju wyrażenia warunkowego będzie generatorem, to relacja może nie zachodzić dla pierwszej wartości generatora, ale może się spełnić dla innej. Oto przykład:

if find ("t",s2)>5 then write ("okay")

W tym wypadku, posługując się – jak poprzednio – łańcuchem:

"try backtracking"

operacja **find** wytwarza najpierw wartość 1, w wyniku czego porównanie nie daje pożądanego rezultatu. Wskutek obowiązywania zasady obliczeń ukierunkowanych, podejmuje się działanie generatora **find,** a ten wytwarza kolejny wynik – 9; tym razem warunek zostaje spełniony.

Analiza łańcuchów

Możliwości Iconu dotyczące dopasowywania wzorców, nazywane **analizowaniem łańcuchów** (ang. string scanning), są tego samego rodzaju, co dopasowywanie łańcuchów w języku SNOBOL4:

$expr_1$ **?** $expr_2$

Tutaj $\mathbf{expr_1}$ dostarcza podmiotu, który jest analizowany według reguł określonych w $\mathbf{expr_2}$. Wszelako Icon – odmiennie niż SNOBOL4 – nie używa wzorców. W zamian rozporządza **funkcjami dopasowującymi,** które mogą przemieszczać punkt analizowania w podmiocie i przyjmować jako wartość dopasowane podłańcuchy. W Iconie funkcja **tab(i)** jest podobna do snobolowego wzorca **TAB(i),** a funkcja **move(i)** odpowiada wzorcowi **LEN(i).**

Przy analizowaniu łańcuchów wolno pominąć w funkcji **find,** i innych podobnych, drugi argument – w tym wypadku ich działanie będzie się odnosić do podmiotu. W ten sposób, w przykładzie:

„try backtracking"

zarówno wyrażenie

every write (find("t",s2))

jak i

every write (s2 ? find("t"))

spowoduje wyprowadzenie wartości 1 i 9.

Pozycje łańcuchów odnalezione przez funkcje analizowania łańcuchów mogą być użyte jako argumenty w funkcjach dopasowujących, jak w wyrażeniu:

write (s2 ? tab (find("y")))

które wypisze **tr.**

Podobnie jak ukierunkowanie celem jest częścią ogólnego mechanizmu obliczania wartości wyrażeń[3] Iconu i może być spożytkowane w obliczeniach dowolnego rodzaju, tak dowolna struktura sterowania lub operacja może zostać użyta w analizie łańcucha. Ilustruje to poniższe wyrażenie, dokonujące – na podobieństwo zamieszczonego wcześniej programu w SNOBOLU4 – tabulacji słów:

```
while line := read() do
  line ? while tab(upto(letters)) do
    count[tab(many(letters))] +:=1
```

Wyrażenie **tab(upto(letters))** odpowiada snobolowemu wzorcowi **BREAK (letters),** a **tab(many(letters))** – wzorcowi **SPAN(letters).**

Operacja w rodzaju:

i +:= 1

powoduje zwiększenie wartości **i** (tu: o jeden). Reszta programu (wydruk) ma spore podobieństwo do wersji w języku SNOBOL4. Tym niemniej pewne cechy Iconu czynią rozwiązanie bardziej spójnym. Klucze **&ucase** i **&lcase** reprezentują, odpowiednio, zbiory wielkich i małych liter; w celu uzyskania zbioru potrzebnych liter wykorzystuje się ich sumę. Jeden z wariantów sortowania tabel wytwarza tu listę alternatywnych indeksów (słów) i odpowiadających im krotności, której dalsza obróbka dokonuje się w jednym wyrażeniu.

[2] Nie powodując przy tym błędu (przyp. tłum.).

[3] Warto w tym miejscu zauważyć, że w Iconie nie występuje pojęcie „instrukcji". Każda iconowa konstrukcja przypominająca instrukcję któregoś z typowych języków może być zbadana pod kątem wytwarzania wyniku (wartości) – jest wyrażeniem (przyp. tłum.).

Icon ma wiele innych właściwości. W ich skład wchodzi rozmaitość nowych struktur sterowania powiązanych z użyciem generatorów. Elementy list mogą być udostępniane za pomocą ich pozycji, lecz również można powodować zwiększanie się i zmniejszanie list na podobieństwo stosów i kolejek. Rozbudowany repertuar operacji na łańcuchach umożliwia drobiazgowe ich przetwarzanie, także bez posługiwania się techniką analizy łańcuchów.

```
procedure main()
  letters := &ucase ++ &lcase
  count := table(0)
  while line := read() do
    line ? while tab(upto(letters)) do
      count[tab(many(letters))] +:= 1
  wlist := sort(count,3)
  while write(get(wlist)," : ",get(wlist))
end
```

Program zliczania słów wyrażony w Iconie

Icon został zaimplementowany na wielu różnych komputerach pracujących pod nadzorem systemu Unix, jak również na komputerach VAX-11, wyposażonych w system VMS, oraz na komputerach osobistych, pracujących pod kontrolą systemu operacyjnego MS-DOS[4]. Wszystkie realizacje Iconu są powszechnie dostępne – nie podlegają sprzedaży.

Tłumaczył i opracował:
ZDZISŁAW PŁOSKI

[4] Realizację Iconu dla systemu operacyjnego MS-DOS na mikrokomputery oparte na mikroprocesorach rodziny 8086/88/186/156 – Icon 6.3 – można otrzymać także w Instytucie Informatyki Uniwersytetu Wrocławskiego za zwrotem kosztów dystrybucji (przyp. tłum.).

LITERATURA

[1] Griswold R. E., Griswold M. T.: Icon Programming Language. Prentice-Hall, Englewood Cliffs (NJ), 1983

LITERATURA UZUPEŁNIAJĄCA

[1] Gizbert-Studnicki P., Karczmarczuk J.: Snobol4. WNT, Warszawa 1984
[2] Karczmarczuk J.: Język programowania Icon. Informatyka, nr 2-3, 4, 5, 1986
[3] Płoski Z.: Słowo o Iconie. O książce „The Icon Programming Language". Instytut Informatyki Uniwersytetu Wrocławskiego, kwiecień 1984.

Ceny ogłoszeń

Od 1 lipca 1987 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia

- ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona – 50 tys., 3/4 str. – 45 tys., 2/3 str. – 40 tys., 1/2 str. – 35 tys., 1/3 str. – 30 tys., 1/4 str. – 25 tys., 1/8 str. – 20 tys., poniżej 1/8 str. – 200 zł za 1 cm²,
- ogłoszenia drobne (zależnie od liczby słów) jedno słowo – 50 zł

Dodatki do ceny podstawowej:

- za każdy dodatkowy kolor + 30%,
- za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%,
- za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%,
- za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%
- za zamieszczenie ogłoszenia na II i III stronie okładki + 50%

Zniżki

dotyczą ogłoszeń – całkowitych powtórzeń

- za ogłoszenia 3–5-krotne – 10%
- za ogłoszenia 6–10-krotne – 20%
- za ogłoszenia 11-krotne i powyżej – 30%
- za artykuły i wkładki reklamowe wykonane przez zleceniodawcę – 40%
- za biuletyny i bloki reklamowe – 60%

W uzasadnionych wypadkach stosuje się zniżki specjalne dla ogłoszeń nie będących powtórzeniami – za zgodą Dyrektora – Naczelnego Redaktora Wydawnictwa NOT SIGMA. Ponadto Biuro Ogłoszeń świadczy usługi w zakresie wykonywania zdjęć czarno-białych i barwnych oraz nadbitek wkładek reklamowych.

Ceny wkładek

- wkładka 2 str. o formacie A4
  nakład do 500 egz. 20 tys. zł
  nakład 500–1000 egz. 35 tys. zł
- wkładka 4 str. o formacie A4
  nakład 500 egz. 40 tys. zł
  nakład 500–1000 egz. 70 tys. zł

Przy wkładkach o nakładzie powyżej 1000 egz. stosuje się wielokrotność powyższych cen, a dodatki za kolory oblicza się jak dla ogłoszeń.

Ogłoszenia przyjmowane się przez:
Dział Ogłoszeń i Reklamy WCiKT NOT SIGMA
ul. Świętojerska 5/7, 00–236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00–950 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

ELEKTRA
SPÓŁDZIELNIA PRACY
ul. Wąska 8, 81–659 Gdynia
Tel.: 241337, 241169
Teleks: 054520 spe pl.

Oferujemy WIELOWYJŚCIOWY, IMPULSOWY ZASILACZ DO URZĄDZEŃ KOMPUTEROWYCH, o następujących parametrach:

$U_1 = +5$ V, $I_1 = 1{,}5$ A
$U_2 = +8 \div 12$ V, $I_2 = 300$ mA
$U_3 = -6 \div 12$ V, $I_3 = 200$ mA

Zasilacz nasz posiada zabezpieczenie ponadnapięciowe i przeciwzwarciowe.
Informacji udzielamy pisemnie, telefonicznie lub teleksem.
Gwarantujemy konkurencyjne oceny, wysoką jakość i krótkie terminy realizacji zamówień.

EO/510/88

ANDRZEJ TUROWICZ
Ośrodek Informatyki
Urząd Wojewódzki
Wrocław

Lokalna sieć komputerowa D-Link (1)

Rozwój telekomunikacji umożliwił łączenie komputerów za pomocą sieci teleksowej lub telefonicznej, w której przesyłanie informacji odbywa się z szybkością od 50 do 9600 bitów na sekundę. W Polsce szybkość 1200 bitów na sekundę należy uważać za graniczną do celów przesyłania danych łączami telefonicznymi. Większe szybkości wymagają specjalnych zabiegów. Z doświadczeń autora wynika, że zadowalające rezultaty można uzyskać przy transmisji z szybkością do 300 bitów na sekundę.

Istnieją również sieci oparte na innych rodzajach łączy niż telefoniczne. W tej klasie istnieją sieci rozległe terytorialnie oraz sieci lokalne (LAN – ang. local area network). Trudno o precyzyjną definicję sieci lokalnej, ale można uważać, że jest to sieć o niewielkim zasięgu, zwykle nie większym niż 5 km. Zasięg terytorialny sieci opartej na łączach telekomunikacyjnych jest w zasadzie nieograniczony i może obejmować setki kilometrów. W sieciach lokalnych zmniejszenie zasięgu umożliwia złagodzenie wymagań dotyczących środków transmisji oraz zwiększenie szybkości transmisji. Sieci lokalne dla komputerów IBM PC/XT lub AT umożliwiają transmisję z szybkościami od 100 tysięcy do 10 milionów bitów na sekundę.

WYBRANE SIECI LOKALNE

Na ogół sieci lokalne są tworzone do połączenia komputerów wewnątrz jednej organizacji gospodarczej. Nic jednak nie stoi na przeszkodzie, aby łączyć komputery organizacji ze sobą współpracujących, jeżeli istnieje konieczność transmisji informacji. Jedynym ograniczeniem jest podana przez producenta długość sieci oraz liczba komputerów, które można dołączyć. Producenci sieci komputerowych zapewniają możliwość pracy na odległość od 1 do 5 km. Producent sieci określa:
- dopuszczalną długość sieci,
- liczbę komputerów w sieci,
- szybkość transmisji.

Poniżej przedstawiono cztery rodzaje sieci dla komputerów IBM PC/XT lub AT. Sieci te mają architekturę magistrali, tj. komputery są dołączane do jednego wspólnego kabla. Informacje techniczne podane w tych opisach są orientacyjne; każdorazowa dostawa lub zakup muszą być poprzedzone informacją od dostawcy lub producenta sieci.

- **Sieć SOFT-Link lub LAN-Link**

Sieć umożliwia połączenie kilku lub kilkunastu komputerów, na odległość około 100 m, za pomocą pary przewodów. Łączność jest zrealizowana wyłącznie środkami programowymi. Transmisja odbywa się przez port RS232. Szybkość transmisji zwykle około 0,1 Mb/s (zależna od producenta oprogramowania).

- **Sieć Transnet**

Jest to jedna z najstarszych i najbardziej popularnych sieci na świecie. Umożliwia połączenie 250 komputerów na odległość około 1,2 km. Łączność jest realizowana za pomocą specjalnej karty sieciowej. Szybkość transmisji 1Mb/s. Magistralę sieci stanowi para skręconych przewodów. Wymaga wzmacniaczy (repetytorów) co około 300 m.

- **Sieć D-Link**

Sieć stanowi rozwinięcie sieci Transnet. Łączność jest zrealizowana za pomocą karty sieciowej. Parametry oraz wymagania są takie same jak dla sieci TRANSNET.

- **Sieć Ethernet**

Jest to najszybsza i najdroższa sieć, o łączności realizowanej przez kartę sieciową. Szybkość transmisji – około 10 Mb/s, kabel współosiowy. Koncepcyjnie taka sama jak Ethernet, lecz znacznie prostsza jest sieć Shina Net. Zasadnicze różnice to brak procesora sieciowego na karcie oraz zmniejszenie zasięgu do 1200 metrów.

PARAMETRY TECHNICZNE SIECI D-LINK

Podstawowym elementem sieci komputerowej D-Link jest karta przeznaczona do montowania w obudowie komputera IBM PC. Na tylnej ściance komputera znajdują się dwa gniazda karty, umożliwiające dołączenie komputera do sieci. Wybór gniazda zależy od rodzaju wtyczki zamontowanej na kablu, stanowiącym przyłącze do magistrali. Kabel dostarczany przez producenta ma zwykle od 2 do 3 metrów długości, lecz można wykonać go we własnym zakresie. Na karcie znajdują się m. in. trzy przełączniki.

- **Przełącznik wyboru numeru komputera** (znaczony SW1 na karcie sieciowej).

Każdy komputer w sieci ma swój numer z przedziału 0–254, określany przez użytkownika sieci. Niedopuszczalne jest włączenie do sieci komputerów o takich samych numerach.

- **Przełącznik typu komputera** (oznaczony symbolem SW2 na karcie);

Pozycja przełącznika zależy od typu komputera, mikroprocesora oraz szybkości pracy, innej dla komputerów AT, XT oraz TURBO. W zasadzie należy żądać od dostawcy sieci lub dostawcy komputera zainstalowania karty i właściwego ustawienia tego przełącznika.

- **Przełącznik terminatorów magistrali** (oznaczony na karcie symbolem TRM).

Można za jego pomocą włączyć lub wyłączyć rezystory kończące magistralę. Można zastosować rezystory umieszczone na kartach skrajnych komputerów. Producent umieścił rezystory na każdej karcie, należy więc zadbać o to, aby w skrajnych komputerach rezystory były włączone, a w pozostałych komputerach wyłączone. Można zrezygnować z rezystorów na kartach i umieścić je na kablu w specjalnych gniazdach.

Według danych producenta długość kabla magistrali może wynosić najwyżej 4000 stóp, tzn. około 1220 m. Co 300 m jest wymagany wzmacniacz-repetytor. Oczywiście wymagania te mają charakter przybliżony; istnieją udane instalacje, pracujące bez wzmacniaczy. Zapewnienie sobie prawidłowej pracy jest możliwe po spełnieniu warunków określonych przez producenta sieci. Szybkość transmisji wynosi milion bitów na sekundę, co dla wielu zastosowań wystarczy z dużym zapasem. Sieć jest zaopatrzona w układy wykrywania transmisji oraz wykrywania kolizji. Programista oraz program są zwolnieni od konieczności badania stanu zajętości sieci. Wszelkie kolizje są wykrywane, a powtórzenia transmisji wykonywane przez karty sieciowe, bez interwencji komputerów i programów.

Istnieje możliwość dołączenia do sieci komputerów bez pamięci dyskowych; komputery takie stanowią tylko terminale. Karta sieciowa w chwili włączania takiego komputera do sieci pobiera system operacyjny z innego komputera i ładuje go do swojego. Wymagany jest jednak odpowiedni program ładujący, dostarczany za dodatkową opłatą przez producenta sieci w postaci układu pamięci ROM. Na płycie karty sieciowej znajduje się pusta podstawka przeznaczona do tego celu.

FUNKCJE KOMPUTERÓW SIECI

Komputer dołączony do sieci może wykonywać swoje zadania lokalne lub współpracować z innymi komputerami poprzez sieć. Może być kilka komputerów spełniających daną funkcję, a także każdy komputer może spełniać kilka funkcji w tym samym czasie.

W sieci D-Link komputer może wykonywać następujące funkcje:

- Przechowywanie plików;

Komputer wykonujący tę funkcję zwany usługodawcą (ang. server) nie musi mieć klawiatury ani monitora ekranowego. Spełnia on rolę pamięci dyskowej dla innych komputerów pracujących w sieci.

- Narzucanie czasu innym komputerom w sieci (zegar);

Z komputera pełniącego tę funkcję inne komputery sieci pobierają czas dzienny, bieżącą datę.

- Gromadzenie na dysku stałym wydruków z innych komputerów;

Kolejki wydruków oraz rozdział wydawnictw i wykorzystanie drukarek są organizowane centralnie.

- Symulowanie na dysku stałym stacji dyskietek dla innych komputerów sieci;
- Korzystanie przez sieć z urządzeń innych komputerów, takich jak drukarki, modemy, stacje dysków stałych lub elastycznych oraz stacje symulowane na dyskach stałych innych komputerów.

Możliwość zdalnego dostępu do zbiorów danych powoduje decentralizację przetwarzania oraz stwarza możliwość pracy wielokomputerowej. Tak pomyślana sieć stanowi logiczne przeciwieństwo wielodostępnej pracy komputera z terminalami. Sieć D-Link nie wyklucza stosowania terminali do każdego komputera. Filozofia sieci D-Link zaleca jednak – zamiast terminali komputera – dołączanie następnych komputerów (np. bez stacji dyskowych) jako umownych terminali sieciowych. Pod względem kosztów rozumianych całościowo, rozwiązanie to nie jest wcale droższe. Należy oczywiście zrozumieć potrzeby wynikające ze specyfiki przetwarzania danych konkretnego użytkownika; mikrokomputer osobisty nie może być traktowany jako jednostka centralna wielkiego wielodostępnego systemu dużego przedsiębiorstwa. Uważam, że bardziej celowe jest dołączenie zamiast terminali kolejnych komputerów do sieci.

INSTALOWANIE SIECI

D-Link dopuszcza pracę w sieci komputerów zaopatrzonych w różne systemy operacyjne (np. DOS lub Xenix) lub w różne wersje danego systemu. Autor sprawdził działanie sprzętu i oprogramowania sieci na komputerach IBM PC/XT TURBO z systemem operacyjnym DOS 3.0 i DOS 3.10

W systemie DOS 3.10 w celu przystosowania komputera do pracy sieciowej należy wykonać czynność logicznego dołączenia karty sieciowej do systemu DOS oraz logicznego dołączenia karty do sieci. Uaktywnienie karty sieciowej uzyskuje się przez wprowadzenia w pliku CONFIG.SYS polecenia zapisu:

DEVICE = D-LINK.DRV /HC

Zapis ten informuje DOS, że urządzenie D-Link jest obsługiwane przez program sterujący D-LINK.DRV. Od tego czasu karta sieciowa może być używana przez DOS. Inne systemy operacyjne wymagają odpowiedniego programu sterującego, dostarczonego przez dostawcę systemu.

Pełna specyfikacja programu D-LINK jest następująca:

D-LINK.DRV [/Dm] [/Pn] [/Mk] /HC

gdzie: **m** oznacza liczbę dysków obsługiwanych przez DOS, **n** – liczbę drukarek, **k** – liczbę modemów.

Program D-LINK.DRV jest dostarczany na dyskietce. Należy zwrócić uwagę na to, by był on akceptowany przez wersję DOS eksploatowaną w danym komputerze. Na dyskietce znajdują się również inne programy usługowe sieci oraz prosty program NET obsługujący sieć, przeznaczony dla początkujących użytkowników. Program ten wywołuje się poleceniem NET. Programy usługowe sieci mogą być wywoływane w trojaki sposób:

- z klawiatury,
- z programu NET,
- z pliku poleceń AUTOEXEC.BAT.

Producent zostawia wybór użytkownikowi sieci, należy jednak przyjąć, że sieć prawidłowo zaprojektowana i zestawiona pod kątem zadań konkretnego użytkownika winna być uaktywniana z programu AUTOEXEC.BAT, tj. programu rozpoczęcia sesji. Pracownicy korzystający z komputerów na swoich stanowiskach pracy nie powinni odczuwać dołączenia ich komputerów do sieci. Każdy komputer powinien mieć tak zaprojektowaną procedurę rozpoczęcia sesji, aby użytkownicy mieli złudzenie pracy indywidualnej. Posługiwanie się programem NET oraz programami usługowymi należy pozostawić programistom. Komputer oddany do eksploatacji pracownikom magazynu lub działu kadr winien wszystkie procedury sieciowe wywoływać w sposób niewidoczny dla obsługi.

PODZIAŁ DYSKÓW STAŁYCH

Dyski stałe mogą być podzielone w różny sposób, w każdym komputerze inaczej. Część pojemności może być przeznaczona na pamięć danego komputera, a część na wirtualne dyskietki dostępne przez sieć dla innych komputerów. Takie dyskietki są nazywane **woluminami.** Na dysku stałym komputera jest rezerwowany obszar pamięci, traktowany przez sieć jak dyskietka. Wolumin nie jest dostępny normalną drogą, lecz tylko przez sieć. W szczególności żadne polecenie (nawet polecenie FORMAT) ani program pracujący w lokalnym komputerze nie mogą uzyskać dostępu do tej części dysku.

Do zarezerwowania odpowiedniej części dysku stałego na woluminy służy polecenie FDISK, które umożliwia zarówno utworzenie strefy (ang. partition) dysku przeznaczonej na DOS, jak i strefy przeznaczonej na woluminy.

Polecenie ma cztery opcje:

- tworzenie strefy DOS,
- zmiana aktywnej strefy,
- skasowanie strefy DOS,
- wyświetlenie podziału dysku.

Wybierając pierwszą opcję uzyskuje się informację o liczbie cylindrów dysku oraz podejmuje się decyzję o tym, ile cylindrów ma zawierać strefa przeznaczona dla systemu DOS. Pozostałe cylindry mogą być użyte przez sieć D-Link na woluminy.

Oczywiście nie jest obowiązkowe wyznaczanie części dysków na woluminy i należy to czynić jedynie w następujących wypadkach:

- gdy w sieci pracują komputery bez stacji dyskowych; wówczas woluminy będą ich „prywatnymi dyskietkami",
- gdy w sieci pracują komputery korzystające ze wspólnych zbiorów tylko w celu odczytania,
- w innych wymaganych przez użytkownika i uzasadnionych wypadkach wymagających wirtualnych dyskietek.

Podział dysków stałych na woluminy należy bardzo dokładnie zaprojektować oraz przemyśleć rozmieszczenie plików w sieci. Niestaranny proces projektowania może się zemścić małą sprawnością przesyłania informacji. Sieć można naprawdę skutecznie zablokować zlecając jej bez wyraźnej potrzeby wielomegabajtowe transmisje zbiorów, które mogłyby być zlokalizowane w tym komputerze, który je przetwarza. Z drugiej strony, w uczelnianej sieci komputerowej jest celowe przechowywanie kompilatorów lub programów w jednym tylko komputerze. Ochronę takiego oprogramowania gwarantuje jedynie system woluminów.

Należy wyraźnie zaznaczyć, że sieć D-Link, sama w sobie zawierająca mechanizmy ochrony przed niepowołanym dostępem do danych i programów, nie chroni ich w sposób dostateczny. Należy zastosować metodę kombinowanego zabezpieczenia informacji w sieci. Dobre rezultaty daje zastosowanie mechanizmów ochrony D-Link we współpracy z mechanizmami ochrony systemu DOS, takimi jak uniemożliwienie zapisu do określonych plików lub udostępnianie jedynie wybranych skorowidzów. W dobrej sytuacji są użytkownicy oprogramowania pracującego w systemie dBase III Plus. Zastosowanie mechanizmów ochronnych gwarantuje odpowiednie prawa dostępu do plików, a nawet do poszczególnych pól w rekordach. W dBase III Plus jest możliwe stosowanie haseł oraz tablic z prawami dostępu do pól w rekordach. Oprócz bardzo wydajnej pracy przy przetwarzaniu danych, dBase III Plus gwarantuje całościową ochronę informacji w skali przedsiębiorstwa. Autor od około roku eksploatuje oprogramowanie oparte na dBase III Plus i uważa, że można ten program polecić jako bardzo sprawne i wydajne narzędzie tworzenia oprogramowania i przetwarzania informacji. Mechanizmy ochrony dBase III Plus w połączeniu z mechanizmami systemu DOS oraz sieci D-Link gwarantują pełną ochronę informacji.

PROGRAMY USŁUGOWE

Podział stref „nie-DOS" na woluminy oraz przydzielenie ich do sieci umożliwiają dwa programy usługowe – VMGR i UMGR. Przed ich omówieniem należy wprowadzić pojęcie numeru komputera oraz numeru grupy użytkowników i numeru indywidualnego użytkownika komputera pracującego w sieci. **Numer komputera** określa fizyczny numer komputera w sieci. **Numer grupy użytkowników** określa logiczny numer systemu eksploatowanego w sieci, np. kadry, płace. **Numer indywidualny** określa numer użytkownika sieci, np. dyrektor, kierownik magazynu, planista.

Jeżeli numer grupy lub użytkownika nie jest określony, to system automatycznie przyjmuje numer 255. Jako zasadę przyjęto, że prawa dostępu do urządzeń, a więc do drukarki lub pamięci dyskowej w komputerze o określonym numerze, są przyznawane konkretnym grupom użytkowników lub użytkownikom indywidualnym. Numer 255 oznacza, że urządzenie jest dostępne dla wszystkich.

Program VMGR (ang. volume manager) dzieli wolną od systemu DOS część dysku stałego na woluminy. Pracuje w trybie konwersacyjnym. Za pomocą klawiszy ze strzałkami wybiera się numer woluminu. Po wybraniu można skasować wolumin klawiszem F2 (system zadaje pytanie Y/N). Ciekawostką jest to, że duże litery wybiera się tu za pomocą klawisza ALT, co ma zapobiec przypadkowemu skasowaniu woluminu. Klawiszem F1 można utworzyć nowy wolumin. Liczba wolnych kilobajtów jest wyświetlana w prawym rogu ekranu. Dyskietki mogą mieć pojemność do 360 KB. System kolejno pyta:
- o pojemność dyskietki w kilobajtach,
- czy dyskietka jest prywatna czy publiczna,
- jeżeli jest prywatna, to system pyta o numer grupy i numer indywidualny użytkownika,
- czy dyskietka jest przeznaczona tylko do odczytu,
- czy zapisać na niej DOS.

Ponieważ w sieci eksploatowanej przez autora nie ma komputerów bez dyskietek, więc nie ma potrzeby zapisywania systemu DOS na woluminach. Dysk stały jest dzielony na woluminy w zasadzie tylko raz. Każda zmiana podziału może doprowadzić do utraty informacji. Dlatego zmiany należy robić po uprzednim skopiowaniu zawartości dysku i woluminów.

Program VMRG wykonuje jeszcze jedną funkcję. Jest nią inicjowanie obszaru przeznaczonego na woluminy. Należy ją wykonać tylko raz, na początku, posługując się klawiszem F3.

Drugim programem usługowym, umożliwiającym komputerom korzystanie z woluminów, jest program UMGR (ang. user manager). Każdy komputer, który na swoim dysku ma założone woluminy i chce je udostępnić do pracy w sieci innym komputerom, musi wywołać program UMGR. Program ten określa, który użytkownik będzie mógł korzystać z których woluminów i na podstawie jakiego hasła. Program umożliwia wstawianie użytkowników na listę oraz usuwanie ich z listy uprawnionych do korzystania z woluminów. Każdy komputer może mieć swoją listę woluminów oraz listę uprawnionych do pracy z nimi. Skrajnym wypadkiem jest możliwość udostępnienia woluminu sobie samemu. W ten sposób zapisuje się informacje na woluminach przeznaczonych tylko do odczytywania.

Program UMGR uruchamia się tylko w wypadku zmiany listy uprawnień dostępu do woluminów. Inne urządzenia sieć D-Link traktuje inaczej. Zezwolenie na dostęp do drukarki, modemu lub stacji dysków, nie będącej woluminem, musi być wznowione za każdym razem, gdy komputer włącza się do sieci. Służy do tego program NET lub SETDEV. Natomiast przyłączanie zdalnych urządzeń odbywa się za pomocą programu CONNECT. Zarówno woluminy, jak i urządzenia zdalne muszą być przyłączone po każdym dołączeniu komputera do sieci. Dołączanie komputera do sieci odbywa się za pomocą programu LOGON (tzn. zarejestrowanie się), a odłączanie – za pomocą programu LOGOFF (tzn. wyrejestrowanie się). Dołączanie komputera do sieci umożliwia także program NET. Należy rozważyć włączenie powyższych programów do pliku AUTOEXEC.BAT. Użytkownik wcale nie musi wiedzieć o wszystkich operacjach, jakie się odbywają w jego komputerze. Włączanie się do pracy z numerem grupy i użytkownika [255,255] nie wymaga podania hasła. Wystarczy napisać:

LOGON nazwa-zadania

Na przykład, po napisaniu:

LOGON magazyn

komputer zostanie automatycznie włączony do sieci. Zadanie będzie miało nazwę MAGAZYN, numer grupy 255, numer użytkownika 255, a prawa dostępu tylko do tych zdalnych urządzeń, które zostały zadeklarowane jako dostępne dla wszystkich (czyli dla grup i użytkownika [255,255]). Włączenie komputera o innym numerze indywidualnym jest dozwolone tylko wówczas, gdy na dysku lub dyskietce jest zapisany plik LOGON.USR. Zawiera on numer grupy, numer użytkownika i jego hasło. Plik ten jest tworzony przez program ALC. Pierwotną nazwą użytkownika programu ALC jest MANAGER, a pierwotnym hasłem – SYSTEM. Stan ten można zmienić. Program ALC po wywołaniu pyta o nazwę użytkownika i hasło. Prawidłowa odpowiedź uprawnia do zmiany hasła lub utworzenia pliku LOGON.USR, przez podanie numerów i hasła użytkownika. Od tej pory polecenie LOGON będzie pytać o hasło. Można mieć kilka dysków lub dyskietek z plikami LOGON.USR i włączać się do sieci pod różnymi nazwami lub z różnymi numerami. Zależy to od projektanta sieci. Komputery mające dostęp do ważnych i poufnych plików muszą być chronione przez hasła zawarte w LOGON.USR, które należy dość często zmieniać.

Nowe książki

WYDAWNICTWA NAUKOWO-TECHNICZNE

Jeffrey D. Ullman „Systemy baz danych". Warszawa 1988, Wydanie I, nakład 5800 egz., s. 504, cena 600 zł

Jest to polski przekład książki pt. „Principles of Database Systems" 2 nd Ed., opublikowanej do rozpowszechnienia w języku polskim na całym świecie, za zgodą wydawnictwa Computer Science Press International, Inc. 1803 Research Boulevard, Rockville, Maryland, 20850 USA, będącego właścicielem wszelkich praw do publikacji i sprzedaży.
Tematem książki są podstawowe zagadnienia projektowania systemów baz danych oraz zasady ich działania. Omówiono fizyczną reprezentację danych, przedstawiono i porównano trzy modele danych (relacyjny, sieciowy i hierarchiczny) wraz z odpowiednimi językami: definicji danych i językami zapytań. Odrębne rozdziały są poświęcone ochronie baz danych, sterowaniu współbieżnymi operacjami w bazie danych oraz systemem rozproszonym. Każdy rozdział jest zakończony zestawem zadań. Bogata bibliografia. Książka jest przeznaczona dla programistów, projektantów systemów przetwarzania informacji, pracowników nauki zajmujących się informatyką oraz dla studentów kierunków informatycznych.

DARIUSZ CABAN
Instytut Cybernetyki Technicznej
Politechnika Wrocławska

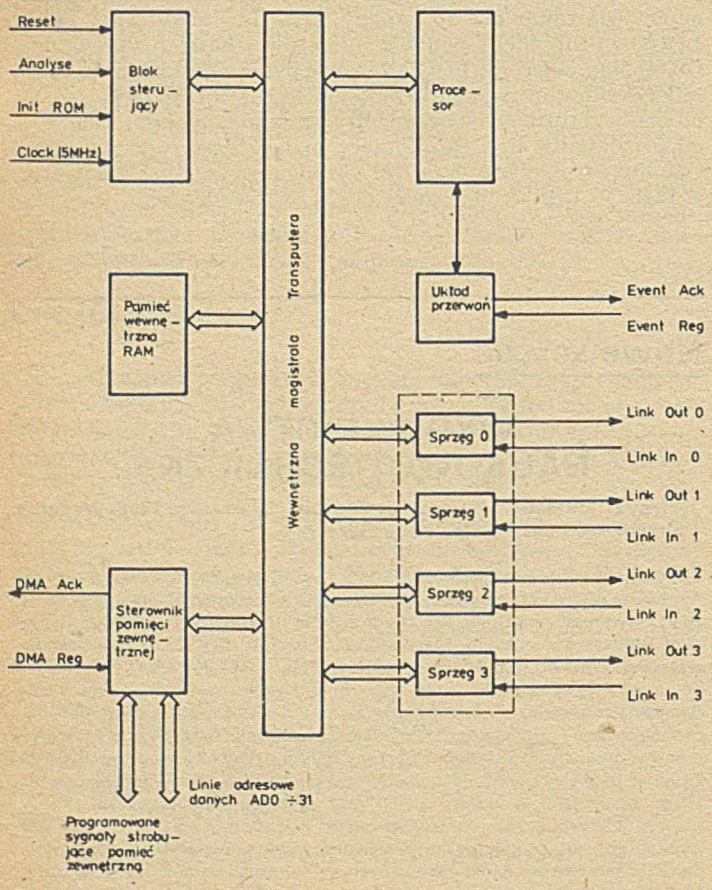
Transputer – narzędzie przetwarzania równoległego

Wśród ogromnej liczby produkowanych obecnie układów scalonych, transputer zasługuje na szczególną uwagę. Układ ten, opracowany przez firmę INMOS z Wielkiej Brytanii, zyskał rozgłos ze względu na zastosowanie nowatorskich koncepcji w organizacji przetwarzania, architekturze i programowaniu. Według zapewnień producenta, na tyle odbiega od mikroprocesora i mikrokomputera jednoukładowego, iż uzasadnione jest używanie nowej nazwy pospolitej – transputer. Transputer jest układem bardzo nowoczesnym, wykonanym w technologii CMOS o szerokości ścieżek 1,5 μm.

Celem artykułu jest przedstawienie właściwości układu, przyjętych nowatorskich idei i rozwiązań, wynikających z dokumentacji technicznej [5, 9]. Jest to tym bardziej celowe, że w prasie fachowej, zagranicznej i krajowej [7], pojawiają się doniesienia, w których faktyczna informacja jest pomieszana ze sloganami reklamowymi.

STRUKTURA I ORGANIZACJA WEWNĘTRZNA

Struktura wewnętrzna transputera jest zbliżona do struktury mikrokomputerów jednoukładowych (rys.). Jednakże, opisane poniżej funkcje realizowane przez poszczególne bloki znacznie odbiegają od układów tradycyjnych.



Struktura wewnętrzna transputera

Procesor

W wersji podstawowej model IMS T414 operuje słowami 32-bitowymi. Lista rozkazów wewnętrznych obejmuje:

- operacje przesyłania, arytmetyczne (m. in. mnożenie i dzielenie), logiczne i przesunięcia,
- typowe operacje sterujące (skoki różnych rodzajów, iteracje i alternatywy),
- operacje na blokach danych i kanałach logicznych,
- operacje na procesach (tworzenie i kończenie procesów, synchronizacja, uaktywnianie i blokowanie).

W strukturze wewnętrznej procesora są umieszczone dwa zegary przyrostowe, które mogą być testowane i ustawiane przez procesy.

Pamięć wewnętrzna RAM

Jest to pamięć o bardzo krótkim czasie cyklu (12,5 ns), pozwalająca na bezkolizyjny dostęp procesora oraz czterech sprzęgów wejściowo-wyjściowych. Pojemność pamięci jest niewielka – 2 KB.

Sprzęgi wejściowo-wyjściowe

Sprzęgi umożliwiają łączenie transputerów w układy wieloprocesorowe. Informacja jest przesyłana szeregowo, asynchronicznie, z potwierdzeniem każdego bajtu. Każdy szereg zapewnia w pełni niezależną transmisję w dwóch kierunkach. Protokół transmisji jest nietypowy, a jej szybkość jest stała i wynosi 10 Mb/s lub 20 Mb/s.

Układ przerwań

Układ przerwań działa inaczej niż w systemach mikroprocesorowych. Oprócz wejścia przerywającego jest wyposażony w wyjście, przez które transputer potwierdza przyjęcie przerwania obsługi. Odmienność obsługi polega na tym, że przerwania odblokowują pewne procesy (realizowane przez transputer), a nie powodują skoku w programie. Przerwanie jest więc traktowane analogicznie do zdarzeń (ang. event) w wieloprogramowych systemach operacyjnych.

Sterownik pamięci zewnętrznej

Umożliwia on umieszczenie dodatkowych układów pamięci w przestrzeni adresowej transputera (maksymalnie 4 GB). Sterownik tworzy sygnały strobujące do zapisywania, odczytywania i odświeżania różnych rodzajów pamięci (RAM, ROM, urządzenia zewnętrzne włączone w przestrzeń adresową). W wypadku dołączenia pamięci dynamicznych, sterownik generuje sygnały $\overline{RAS}$, $\overline{CAS}$, $\overline{WR}$, a także sterujące multiplekserem wiersz-kolumna i zatrzaskujące szynę adresową. Sterownik automatycznie włącza cykle odświeżania pamięci, wystawiając 10-bitowy adres. Czasy występowania i trwania sygnałów strobujących są programowane w fazie inicjalizacji transputera.

Blok sterujący

Zapewnia inicjalizację pracy transputera (ang. bootstrap) oraz umożliwia wymuszenie przejścia do trybu „analizy stanu". Blok bada stan wewnętrznego znacznika błędów i odwzorowuje go na jedno z wejść układu (Error), co umożliwia reagowanie na błędy przez inne transputery.

Charakterystyczną cechą architektury transputera jest brak dostępu do jego magistrali wewnętrznej. Główne funkcje magist-

rali – dołączanie portów i równoległych procesorów – są realizowane za pośrednictwem sprzęgów wejściowo-wyjściowych. Dzięki dużej szybkości transmisji, nie ma to istotnego wpływu na przepustowość systemu.

PRZESTRZEŃ ADRESOWA

Transputer operuje 32-bitowymi adresami, co umożliwia zaadresowanie maksymalnie 4 GB. Najniższe adresy odpowiadają pamięci wewnętrznej. Najwyższe adresy są przeznaczone do ewentualnego dołączenia zewnętrznej pamięci ROM. Nie ma żadnych innych ograniczeń na konfigurowanie pamięci zewnętrznej.

Transputery operują słowami wielobajtowymi (32-bitowymi w wypadku IMS T414). Pamięć wewnętrzna i zewnętrzna ma organizację słowną tzn. zapewnia równoczesny dostęp do pełnego słowa w ramach jednego cyklu odczytu lub zapisu. Niemniej, najmniejszą adresowalną jednostką pamięci jest bajt. Najmniej znaczące bity adresu (A0 i A1 w IMS T414) są wskaźnikiem bajtu w ramach słowa. Transputer automatycznie realizuje składanie i wydzielanie bajtów z adresowanych słów pamięci.

Tabela 1. Zarezerwowane obszary w pamięci wewnętrznej transputera

Adres	Przeznaczenie
80000000H	Słowo sterujące kanału wyjściowego, związanego ze sprzęgiem 0
80000004H	Słowo sterujące kanału wejściowego, związanego ze sprzęgiem 0
80000008H	Słowo sterujące kanału wyjściowego, związanego ze sprzęgiem 1
8000000CH	Słowo sterujące kanału wejściowego, związanego ze sprzęgiem 1
80000010H	Słowo sterujące kanału wyjściowego, związanego ze sprzęgiem 2
80000014H	Słowo sterujące kanału wejściowego, związanego ze sprzęgiem 2
80000018H	Słowo sterujące kanału wyjściowego, związanego ze sprzęgiem 3
8000001CH	Słowo sterujące kanału wejściowego, związanego ze sprzęgiem 3
80000020H	Słowo sterujące kanału wejściowego, związanego z sygnałem przerwań transputera
80000024H 80000047H	Obszar roboczy sprzętowego jądra wieloprogramowego zawierającego m. in. wskaźniki kolejek procesów

W ramach pamięci wewnętrznej, część adresów ma stałe przypisane funkcje. Mapę zarezerwowanego obszaru przedstawiono w tabeli 1. Przy analizie tej tabeli należy brać pod uwagę, że adresy transputera są reprezentowane w postaci liczb ze znakiem w kodzie uzupełnieniowym. Z tego względu najniższym adresem jest liczba # 80000000H (w zapisie szesnastkowym).

Transputer podczas wstępnej inicjalizacji pobiera informacje dotyczące wymaganego trybu pracy z łącza szeregowego lub z pamięci ROM (jeżeli na wejściu układu oznaczonym InitROM jest wymuszony stan "1"). W tym ostatnim wypadku, transputer traktuje w specjalny sposób adresy większe niż 7FFFFF6CH. W bajtach o adresach 7FFFFF6CH-7FFFFFF8H powinna być umieszczona informacja o wymaganiach czasowych dotyczących sterownika pamięci zewnętrznej, a bajty 7FFFFFFFEH-7FFFFFFFH powinny zawierać adres początku programu inicjującego transputer.

WSPÓŁBIEŻNE WYKONYWANIE PROCESÓW

Najistotniejszą cechą, różniącą transputer od sprzętu mikroprocesorowego jest zorganizowanie wieloprogramowości na poziomie wewnętrznej struktury układu. Wiele funkcji, tradycyjnie wykonywanych przez jądro wieloprogramowych systemów operacyjnych, jest realizowanych sprzętowo w wypadku transputera. Do funkcji takich należą:

- szeregowanie procesów (ang. scheduling),
- synchronizacja i komunikacja między procesami,
- kontrola upływu czasu procesowego.

Szczególną zaletą wieloprogramowości zrealizowanej sprzętowo jest zminimalizowanie czasu wykonania funkcji szeregujących. Przykładowo, długość kolejek procesów gotowych i zablokowanych nie wpływa w ogóle na czas wykonywania procesu aktywnego. Również uwolnienie programisty systemowego od konieczności tworzenia podstawowych funkcji jądra jest atrakcyjne – oprogramowanie systemowe staje się dzięki temu prostsze i mniej narażone na błędy synchronizacji.

Wadą rozwiązania jest narzucona, sztywna struktura jądra systemu operacyjnego transputera. Współbieżne procesy mają w nim dwa poziomy priorytetu. Procesy o niższym priorytecie (oznaczonym jako 1) są wykonywane z podziałem czasu, według algorytmu karuzelowego (ang. round-robin). Każdy niezablokowany proces o tym priorytecie jest wykonywany w ciągu sztywno określonego kwantu czasu, po którym przechodzi na koniec kolejki procesów oczekujących.

Procesy o wysokim priorytecie (0) przerywają wykonywanie procesów z niższym (wywłaszczają je) natychmiast po odblokowaniu. Są wykonywane bez podziału czasu, tzn. pozostają aktywne aż do ponownego zablokowania lub zakończenia.

Ciekawym rozwiązaniem, zastosowanym w transputerach, jest dopuszczenie do zmiany procesu aktywnego jedynie w trakcie wykonywania niektórych rozkazów (skoku bezwzględnego i iteracji). Dzięki temu, przez odpowiedni dobór instrukcji, można tworzyć obszary na pewno wykonywane łącznie, co znacznie upraszcza synchronizację.

Jedynym zaimplementowanym w transputerach mechanizmem synchronizacji jest wymiana komunikatów przez kanał logiczny (zgodnie ze sformułowaną przez C.A.R. Hoare'a koncepcją CSP [3]). Mechanizm ten polega na zablokowaniu procesu nadającego lub odbierającego komunikat do czasu, aż oba (nadający i odbierający) są gotowe do transmisji. Następnie, po przesłaniu komunikatu, procesy te zostają odblokowane.

Jednolity mechanizm kanałów logicznych jest wykorzystywany do:

- przesyłania komunikatów i synchronizacji procesów w obrębie jednego transputera,
- przesyłania komunikatów przez sprzęgi wejściowo-wyjściowe do procesów realizowanych przez inne transputery (8 kanałów logicznych o adresach podanych w tab. 1),
- odbioru sygnałów przerwania (1 kanał wejściowy),
- uzależnienia procesów od upływu czasu rzeczywistego (sygnały osiągnięcia przez wewnętrzne zegary przyrostowe zadanego stanu są traktowane programowo jako kanały logiczne).

LISTA ROZKAZÓW I PROGRAMOWANIE

Transputery są przykładem sprzętu obliczeniowego, projektowanego pod kątem efektywnego wykonywania programów pisanych w określonym języku wysokiego poziomu – konkretnie, w języku Occam (por. [1]). Według materiałów firmowych w transputerze jest osiągana bardzo duża efektywność programów wynikowych (uzyskanych po kompilacji z językiem Occam), porównywalna z programami pisanymi na poziomie języka assemblerowego [9]. Z tego względu producent zakłada, że użytkownicy nie korzystają z języka symbolicznego (nawet przy tworzeniu oprogramowania systemowego). Konsekwencją takiego nastawienia jest na przykład podawanie czasów realizowania przez transputer instrukcji języka Occam, zamiast rozkazów wewnętrznych.

Pomimo takiego ukierunkowania transputera nie wykonuje on bezpośrednio programów pisanych w języku Occam (co sugerują niektóre publikacje). Lista rozkazów transputera została opracowana zgodnie z koncepcją RISC (komputerów o zredukowanej liście rozkazów). Większość rozkazów jest wykonywana w pojedynczym cyklu zegara, co umożliwia osiąganie bardzo dużej szybkości przetwarzania (zależnie od wersji transputera od 5 do 10 MIPS). Dodatkowo, przy opracowywaniu listy rozkazów przyjęto, że musi być ona jednolita dla transputerów o różnych długościach słowa maszynowego.

Rozkazy transputera operują następującymi rejestrami (wszystkie mają długość słowa maszynowego, tzn. 32 bity w wypadku IMS T414):

Iptr (Instruction Pointer) – licznik rozkazów,

Wptr (Workspace Pointer) – rejestr bazowy, ustalający obszar aktywnego procesu, używany do wyznaczania efektywnego adresu zmiennych lokalnych i kanałów,
Oreg (Operand Register) – rejestr argumentu, w którym jest formowany efektywny argument wykonywanego rozkazu,
Areg, Breg, Creg – rejestry robocze, ułożone w stos, tak że dostępny jest zawsze pierwszy; operacja ładowania do Areg automatycznie powoduje przesunięcie zawartości z Breg do Creg i z Areg do Breg; podobnie przy pobraniu argumentu ze stosu. Zawartość rejestru Breg jest przypisywana do Areg a Creg do Breg.

Tabela 2. Podstawowa lista rozkazów transputera

Nazwa	Kod	Funkcja
ldc	04H	Ładowanie wartości stałej (z Oreg) do Areg
adc	08H	Dodawanie wartości stałej do Areg
eqc	0CH	Porównanie zawartości Areg ze stałą
ldl	07H	Ładowanie zmiennej lokalnej do Areg (adresowanej jako Oreg przesunięcie względem Wptr)
stl	0DH	Składowanie zawartości Areg do zmiennej lokalnej
ldnl	03H	Ładowanie zmiennej globalnej (adresowanej jako Oreg przesunięcie względem Areg)
stnl	0EH	Składowanie zawartości Breg do zmiennej globalnej
ldlp	01H	Ładowanie do Areg wskaźnika zmiennej lokalnej
ldnlp	05H	Ładowanie do Areg wskaźnika zmiennej globalnej
ajw	0BH	Przesunięcie obszaru roboczego procesu (dodanie zawartości Oreg do wskaźnika Wptr)
j	00H	Skok bezwarunkowy (według adresu z Oreg)
cj	0AH	Skok warunkowy (uzależniony od zawartości Areg)
call	09H	Wywołanie podprogramu (połączone ze składowaniem Iptr, Areg, Breg, Creg na początku obszaru roboczego)
pfix	02H	Ładowanie 4-bitowego argumentu (tzw. prefiksu) na koniec rejestru Oreg
nfix	06H	Ładowanie 4-bitowego prefiksu, połączone z zanegowaniem poprzedniej zawartości Oreg
opr	0FH	Wykonanie funkcji zależnej od zawartości Oreg

Wszystkie rozkazy transputera są jednobajtowe: 4 bity zajmuje kod rozkazu (tabela 2), 4 pozostałe – jego argument. Właściwy argument jest na ogół dłuższy niż 4 bity i jest pobierany z rejestru Oreg. Część argumentowa rozkazu jest umieszczana wcześniej na najmniej znaczących bitach tego rejestru, po przesunięciu jego poprzedniej zawartości. Dzięki rozkazom prefiksującym (pfix, nfix), jest możliwe wcześniejsze przygotowanie zawartości rejestru Oreg. Zależnie od wykonywanego rozkazu zawartość rejestru Oreg wykorzystywana jest jako:
- argument bezpośredni;
- adres lokalny argumentu, tzn. przesunięcie adresu względem zawartości rejestru bazowego Wptr;
- adres globalny argumentu, w którym Oreg zawiera przesunięcie względem zawartości rejestru Areg;
- wskaźnik argumentu lokalnego, na którego podstawie jest tworzony adres pośredni;
- wskaźnik argumentu globalnego;
- adres skoków liczony względem Wptr;
- rozszerzenie kodu rozkazu.

Tabela 3. Lista rozkazów jednobajtowych, wywoływanych przez użycie opr (o postaci opr n, gdzie n jest kodem podanym w tabeli)

Nazwa	Kod	Funkcja
rev	00H	Zamiana zawartości rejestrów Areg i Breg
gcall	06H	Skok pośredni (zamiana zawartości Areg i Iptr)
lb	01H	Ładowanie zawartości bajtu według adresu (wskaźnika) z Areg
add	05H	Dodawanie zawartości rejestrów Areg + Breg z kontrolą przepełnienia
sub	0CH	Odejmowanie Areg-Breg
diff	04H	Odejmowanie bez kontroli pożyczki i przekroczenia zakresu
prod	08H	Mnożenie bez kontroli przepełnienia
gt	09H	Porównanie zawartości Areg i Breg (niszczące)
bsub	02H	Tworzenie wskaźników do tablic bajtowych (bsub)
wsub	0AH	lub słowowych (wsub) według adresu początku tablicy z Areg i indeksu Breg
in	07H	Przesłanie komunikatu o długości według Areg z (do) kanału Breg pod adres Creg
out	0BH	
outbyte	0EH	Przesłanie bajtu z Areg do kanału według Breg
outword	0FH	Przesłanie słowa transputera z Areg do kanału
startp	0DH	Tworzenie procesu o identyfikatorze z Areg
endp	03H	Kończenie (usuwanie) procesu, w którym wystąpił rozkaz (w Areg identyfikator procesu następcy)

W wypadku rozkazów działających na dwóch argumentach (np. przesłania), drugi jest określany domyślnie jako aktualny szczyt stosu rejestrów roboczych Areg. Pełni on funkcję normalnego rejestru roboczego (z którego odczyt jest niszczący).

W zestawie rozkazów korzystających z rejestru Oreg, specjalne znaczenie ma **opr.** Rozkaz ten umożliwa uzupełnienie listy rozkazów transputera – pobiera rozszerzenie kodu z rejestru Oreg, wcześniej zapełnionego rozkazami prefiksującymi. Aktualna lista rozkazów obejmuje rozszerzenia maksymalnie długości 1 bajtu, mimo że teoretycznym ograniczeniem jest pełne słowo maszynowe. W tabelach 3 i 4 podano listy rozkazów z rozszerzeniami 4- i 8-bitowymi.

Rozkazy transputera, wywoływane za pośrednictwem **opr,** nie mogą używać rejestru Oreg jako argumentu właściwego. Argumentami dla realizowanych przez nie działań jest zawartość stosu rejestrów Areg-Breg-Creg. Ogólnie, Areg musi zawierać pierwszy argument operacji, a Breg i Creg – ewentualnie dalsze. Realizowany rozkaz usuwa ze stosu wszystkie wartości wykorzystane jako argumenty, umieszczając wynik na nowym szczycie. Przykładowo, operacja **add** spowoduje następujące ustawienie wartości stosu rejestrów:

Tabela 4. Lista rozkazów o rozszerzonych kodach, wywoływanych sekwencją: pfix [n/16] opr (n mod 16)

Nazwa	Kod (n)	Funkcja
ret	20H	Powrót z podprogramu
ldpi	1BH	Załadowanie zawartości Iptr do Areg
gajw	3CH	Zamiana zawartości Wptr i Areg
lend	21H	Koniec iteracji, tzn. skok względny z dekrementacją licznika
move	4AH	Przesłanie komunikatu w pamięci
sb	3BH	Składowanie bajtu wg adresu pośredniego
bcnt, wcnt	34H,3FH	Przekształcanie indeksu bajtowego na słowowy i odwrotnie
resetch		Inicjalizacja kanału
ldtimer		Przepisanie stanu zegara do Areg
tin		Czytanie z kanału zegara (zawieszenie czasowe procesu)
sethalterr	58H	Ustawienie /zerowanie/
clrhalterr	57H	/testowanie specjalnego trybu
testhalterr	59H	pracy transputera halt-on-error
stoperr	55H	Wymuszenie zakończenia procesu po ustawieniu się znacznika błędów
seterr	10H	Ustawienie znacznika błędów
testerr	29H	Testowanie znacznika błędów
csub0,csub1	13H,4DH	Testowanie poprawności indeksu w strukturze /komunikacie z ustawieniem znacznika błędu
and, or, xor	46H,4BH,33H	Operacje logiczne oraz przesunięcia arytmetyczne zawartości Areg
not, shl, shr	32H,41H,40H	
sum,rem	52H,1FH	Operacje arytmetyczne bez
mul,div	53H,2CH	przeniesień, bez i z ustawieniem znacznika błędu przy nadmiarze
ladd,lsub,lsum	16H,38H,37H	Operacje arytmetyczne wyznaczające przeniesienie
ldiff,lmul	4FH,31H	i uwzględniające je
ldiv	1AH	
lshl,lshr,norm	36H,35H,19H	Operacje przesunięcia na liczbach podwójnych
unpacksn,fmul	63H,72H	Operacje na liczbach zmienno-
roundsn,ldinf	6DH,71H	przecinkowych pojedynczej
postnormsn	6CH	precyzji i postaci zgodnej
cflerr	73H	z normą IEEE
alt,altwt	43H,44H	Rozkazy umożliwiające
talt,taltwt	4EH,51H	zorganizowanie alternatywy
altend	45H	procesów, uzależnionych
enbt,dist	47H,2EH	od operacji kanałowych
enbc,disc	48H,2FH	i zegara
enbs,diss	49H,30H	

Creg ← wartość przypadkowa,
Breg ← poprzednia zawartość Creg,
Areg ← suma poprzednich zawartości Areg i Breg.

Zawartość stosu rejestrów, zależnie od wykonywanych rozkazów, może być wykorzystywana następująco:

- jako argumenty rejestrowe,
- jako znacznik przeniesienia przy operacjach na liczbach o podwójnej precyzji; np. operacja **ladd** dodaje zawartość Areg + Breg + Creg (przeniesienie), wynik umieszcza się w Areg, a ewentualne nowe przeniesienie w Breg,
- jako rejestr bazowy przy adresowaniu zmiennych globalnych,
- jako wskaźnik pamięci przy adresowaniu pośrednim (operacje **lb** i **sb** przesyłają zawartość jednego bajtu między stosem rejestrów a pamięcią adresowaną pośrednio zawartością Areg),
- jako wskaźnik adresu skoków pośrednich,
- jako rejestry bazowe i indeksujące, przy dostępach do struktur danych (przykładowo, **bsub** wyznacza adres pośredni bajtu na podstawie zadanego w Areg adresu bazowego tablicy oraz indeksu z rejestru Breg),
- jako rejestry licznikowe przy operacjach na komunikatach, w pętlach iteracyjnych itp.,
- jako identyfikatory procesów, opisywanych adresami bazowymi obszarów roboczych i priorytetem (np. **startp** używa rejestr Areg, z którego pobiera identyfikator procesu, umieszczając go na końcu łańcucha procesów gotowych, oczekujących na przydział procesora),
- jako identyfikatory kanałów logicznych, tj. adresy ich słów sterujących; np. operacja **output** zawiera w Areg adres słowa sterującego kanału, w Breg adres początku komunikatu, a w Creg licznik jego długości; operacja powoduje przesłanie, jeżeli w słowie sterującym znajduje się identyfikator procesu (w którym wystąpił rozkaz **input**, z tym samym identyfikatorem); w przeciwnym wypadku operacja powoduje jedynie załadowanie identyfikatora aktywnego procesu do tego słowa.

Lista rozkazów transputera umożliwa realizację wszystkich znanych trybów adresowania. Na ogół wymaga to składania po kilka rozkazów, co jest konsekwencją bardzo podstawowego charakteru realizowanych czynności. Jest to w pełni rekompensowane dużą szybkością wykonywania tych rozkazów.

ZASTOSOWANIA TRANSPUTERÓW

Systemy transputerowe są przeznaczone głównie do realizacji zadań przetwarzania równoległego. Decydujące znaczenie mają w tym wypadku następujące cechy architektury:

- Odejście od struktury magistralowej systemu na rzecz struktury rozproszonej (np. drzewiastej, tablicowej itp.) Unika się dzięki temu konfliktów dostępu do magistrali, a w związku z tym konieczności wprowadzania układów arbitrażu oraz strat mocy obliczeniowej przy „nasycaniu się" przeciążanej magistrali.
- Maksymalne uproszczenie technicznej realizacji połączenia między transputerami (przez zminimalizowanie połączeń i brak układów pośredniczących).
- Wprowadzenie sprzętowego jądra wieloprogramowego, zapewniającego współbieżność, kolejkowanie i synchronizację procesów,
- Zapewnienie sprzętowych mechanizmów synchronizacji procesów realizowanych w różnych transputerach, nie różniących się programowo od normalnej synchronizacji międzyprocesowej. Mechanizmy te umożliwiają swobodne rozdzielanie procesów współbieżnych między transputery, bez niebezpieczeństwa powstania błędów synchronizacji uzależnionych od tego rozdziału.
- Umożliwienie zewnętrznego monitorowania i analizy błędów transputera. Dzięki temu system wielu transputerów może reagować na błędy poszczególnych jednostek.
- Osiągnięcie dużej efektywności przetwarzania programów pisanych w języku wysokiego poziomu z mechanizmami współbieżności (język Occam).

Ze względu na ubogie środki synchronizacji procesów, a w szczególności brak możliwości pracy ze wspólną pamięcią, nie wszystkie problemy przetwarzania równoległego są rozwiązywane w pełni efektywnie. Konieczne jest często utrzymywanie tych samych informacji w pamięciach lokalnych wielu transputerów, przy równoczesnym zwiększonym obciążeniu łączy (będącym konsekwencją ciągłego uaktualniania tych informacji).

Sieć transputerów jest idealnie dostosowana do przetwarzania takich zadań, w których procesy są słabo ze sobą powiązane. Szczególnie spektakularne efekty osiągnięto w grafice komputerowej [2] oraz przetwarzaniu obrazów (np. w tomografii [4]).

Na zakończenie warto wspomnieć o dostępnych w handlu systemach transputerowych. Układ jest jeszcze zbyt nowy, aby ich liczba była duża. Co więcej, pewne cechy układu predysponują go do zastosowań militarnych, o czym brak danych. Różni producenci oferują systemy modułowe, zawierające od 1 do 16 transputerów, a także karty do mikrokomputera IBM PC/AT z pojedynczym transputerem [6, 8].

LITERATURA

[1] Błaszczak S.: Język Occam. Informatyka, nr 7, 8, 1988
[2] Bolton M.: The Transputer and Its Application. Microcomputer '86 – design, practice, education. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, nr 73, Wrocław, 1986
[3] Hoare C. A. R.: Communicating Sequential Processes. Prentice Hall, 1985
[4] Kingswood N., et. al.: Image reconstruction using the transputer. IEE Proc. on Computers Digital Techn., Vol. 133, No. 3, May 1986
[5] May D.: Computer Writer's guide preliminary. INMOS Ltd., 1986
[6] New products and services OUG Newsletter, No. 5, July 1986
[7] Transputery – przyszłość techniki komputerowej? Informatyka, nr 6, czerwiec 1986
[8] The Transputer Family, Product Information. INMOS Ltd., 1985
[9] Transputer Reference Manual, INMOS Ltd., 1985.

Komputery optoelektroniczne

W ubiegłym roku na posiedzeniu plenarnym Polskiego Komitetu Optoelektroniki, działającego przy Stowarzyszeniu Elektryków Polskich, przewodniczący Zespołu Roboczego „Optoelektronika informatyczna", doc. dr Zdzisław Wrzeszcz z Instytutu Maszyn Matematycznych w Warszawie wygłosił referat pt. „Komputery optoelektroniczne".

Na wstępie referent wskazał na ograniczenia, jakie występują przy dalszym rozwoju komputerów elektronicznych i możliwości jakie stwarzają układy optyczne. Jednocześnie podkreślił trudności realizacyjne tych układów, tak że po okresie euforii z przełomu lat sześćdziesiątych i siedemdziesiątych, związanej głównie z rozwojem pamięci holograficznych, nastąpił regres, który dopiero ostatnio jest przełamywany.

W rezultacie komputera optycznego wciąż jeszcze nie ma. Prowadzi się prace badawcze nad jego architekturą i elementami, które muszą osiągnąć dostateczną dojrzałość techniczną, aby być konkurencyjne w stosunku do rozwiązań elektronicznych. Najbardziej obiecująca jest technika hybrydowa, choć wymaga ona układów przetwarzania sygnałów elektrycznych na optyczne i odwrotnie.

W dalszym ciągu referatu omówiono architekturę systemów komputerowych z uwzględnieniem specyfiki realizacji optycznych, procesory i układy podstawowe włącznie z procesorami analogowymi i binarnymi, optoelektroniczne pamięci komputerowe, wśród których scharakteryzowano rozwijające się najbardziej dynamicznie dyski optyczne, optyczne urządzenia wejścia-wyjścia z drukarkami laserowymi oraz połączenia optyczne obejmujące zarówno wewnętrzne połączenia komputerów (między elementami i podzespołami) jak i połączenia między komputerami. Końcową część referatu poświęcono roli holografii w komputerach optoelektronicznych, a także pierwszym zastosowaniom i kierunkom badań.

W podsumowaniu wyodrębniono tematy, na których powinny się koncentrować prace nad komputerami optoelektronicznymi. Obejmują one: komputer masowego zastosowania, superkomputery oraz komputery związane z rozwojem sztucznej inteligencji, zarówno z bazą wiedzy, jak i w formie sieci neutronowej.

JR

ANDRZEJ HUZAR
ANDRZEJ KALIŚ
Centrum Obliczeniowe
Politechnika Wrocławska

Sieć komputerowa Jednolitego Systemu SKJS2, wersja 1

Na łamach INFORMATYKI już prezentowano Międzyuczelnianą Sieć Komputerową (MSK) zaprojektowaną na Politechnice Wrocławskiej. Jest to rozległa sieć heterogeniczna, łącząca komputery Odra 1305 i EC 1032. Ten artykuł dotyczy sieci komputerowej Jednolitego Systemu, nazwanej SKJS2, wersja 1. Jest to sieć utworzona wyłącznie ze sprzętu produkowanego lub kompletowanego przez Zakłady Elektroniczne ELWRO we Wrocławiu.

W 1984 r. Centrum Obliczeniowe Politechniki Wrocławskiej rozpoczęło realizację umowy na dostarczenie sieci komputerowej maszyn Jednolitego Systemu. Zespoły wykonawcze podjęły się tego złożonego zadania mimo trwania końcowych prac przy uruchamianiu sieci MSK. O ile prace nad MSK w znacznym stopniu były pracami badawczymi i eksperymentalnymi, a w swym zamierzeniu dopiero przygotowywały kadry specjalistów, o tyle sieć SKJS2 jest produktem handlowym. Sieć SKJS2 była gotowa już w końcu 1986 r., czyli jej realizacja trwała około 3 lat, łącznie z fazami tworzenia koncepcji, projektowania, kodowania i testowania oprogramowania sieciowego. Oprogramowanie to jest obecnie oferowane przez ELWRO-SERVICE.

Geneza projektu

Systemy komputerowe Jednolitego Systemu EC 1032 oraz EC 1034 zapewniają swoim użytkownikom oprogramowanie TELE JS, rozpowszechnione w kraju i poza jego granicami. Umożliwia ono dostęp z terminali do wybranych podsystemów, takich jak System Kontroli i Obsługi Terminali SKOT, i do baz danych lub do podsystemu wielodostępnego TSO. Terminale zdalne (przyłączone do procesora czołowego) są obsługiwane przez program emulacyjny EP umieszczony w tym procesorze. Terminale lokalne, przyłączone bezpośrednio do kanału komputera obliczeniowego, są obsługiwane na poziomie fizycznym metodą TCAM (metoda telekomunikacyjna) lub BTAM. Podstawową wadą oprogramowania TELE JS jest brak standardów jego stosowania. Dlatego, choć jest możliwość łączenia się z terminala z różnymi aplikacjami, rzadko się z tego korzysta (wygenerowanie takiego programu obsługi terminali nie jest zresztą proste).

Idąc za przykładem firmy IBM, która rozpowszechniła swoją architekturę sieci SNA (niezgodnej z modelem ISO), tworząc własne sieci komputerowe (np. EARN), polski przemysł komputerowy wystąpił z inicjatywą zaprojektowania własnej sieci komputerowej, zgodnej z architekturą sieci otwartych.

Główne wymagania i ograniczenia

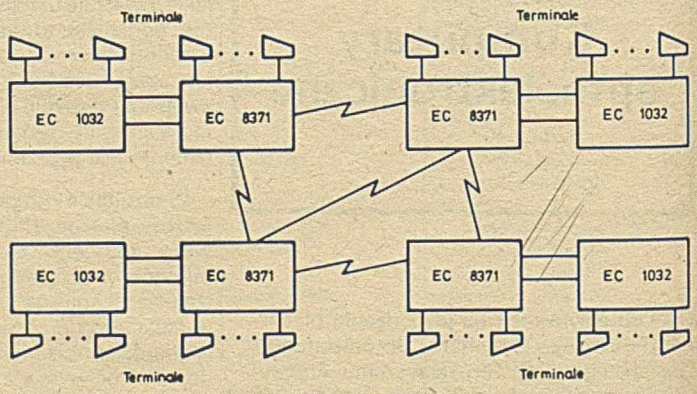
Przystępując do zaprojektowania sieci SKJS2, wersja 1, wykonawca przedstawił wymagania i założenia.

- Wymagania użytkowe zapewniają:
  - komunikację między dowolnym terminalem sieci a dowolnie wybraną aplikacją sieciową;
  - łączność między konsolami operatorskimi systemów autonomicznych (komputerów obliczeniowych);
  - transfer zbiorów sekwencyjnych między dowolnymi komputerami obliczeniowymi sieci.
- Wymagania projektowe zapewniają:
  - możliwość tworzenia dowolnej konfiguracji sieci;
  - możliwość rozbudowy sieci o dodatkowe komputery obliczeniowe, węzły i zasoby (aplikacje, terminale);
  - prostą organizację własnych systemów użytkowych;
  - obsługę w sieci dość dużej liczby terminali.
- Wymagania sprzętowe producenta:
  - sprzęt pracujący w sieci musi być produkowany lub kompletowany przez ZE ELWRO;
  - oprogramowanie dostarczane użytkownikom, pracujące dotychczas pod kontrolą systemu TELE JS (TSO i SKOT), powinno być dostępne w sieci;
- Wymagania eksploatacyjne zapewniają:
  - diagnozowanie stanów sieci;
  - łagodzenie skutków awarii sprzętu;
  - raportowanie i pomiary pracy sieci.

Jednocześnie założono, że terminale przyłączone do komputera czołowego będą obsługiwane przez program NCP (ang. network control program), co pociągało za sobą konieczność zastosowania telekomunikacyjnej metody dostępu TCAM 5.

Struktura sieci

Sieć komputerowa składa się ze środków transmisji danych, komputerów i terminali. Ogólną strukturę sieci przedstawiono na rys. 1. Istotną cechą tej sieci jest połączenie w jednym komputerze EC 8371.01 funkcji komputera czołowego i komputera węzła sieci. Każdy komputer EC 8371.01 jest umieszczony w niewielkiej odległości od komputera obliczeniowego (identycznie jak w TELE JS). Dwa lub więcej komputerów EC 8371.01 jest połączone między sobą modemami i liniami telefonicznymi.



Rys. 1. Przykładowa struktura sieci SKJS2, wersja 1

W sieci można wykorzystywać następujące komponenty sprzętowe:
- komputery obliczeniowe JS z systemem OS/JS 5.01 MVT;

– komputer czołowy i węzła EC 8371.01;
– terminale, jak stacja abonencka EC 8575M, rodzina monitorów ekranowych EC 7900 (7910, 7914, 7915, 7917);
– środki transmisji, tj. dotychczasowe połączenia terminali z EC 8371.01, modemy EC 8013 (lub inne modemy pracujące z szybkością do 9600 bodów) do łączenia EC 8371.01 między sobą.

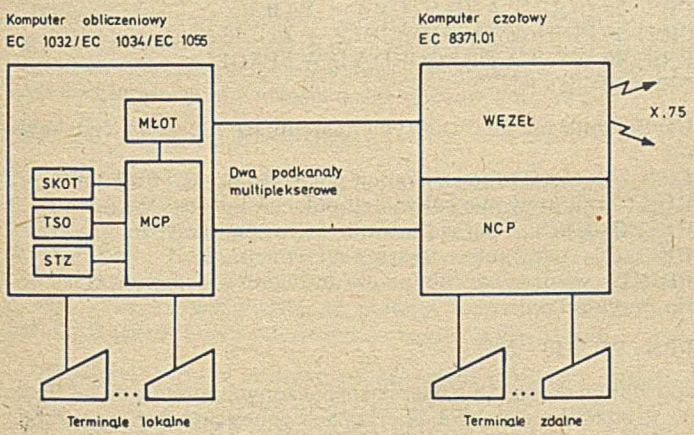
Struktura sprzętowa sieci praktycznie nie różni się od tej, jaką mają użytkownicy w konfiguracjach autonomicznych, potrzebne są tylko dodatkowe modemy i linie między węzłami. Praktycznie bowiem każdy użytkownik EC 1032 dysponuje procesorem EC 8371.01.

Oprogramowanie sieci

System operacyjny OS/JS 5.01 MVT, telekomunikacyjna metoda dostępu TCAM 5 oraz program sterujący NCP stanowią środowisko, w którym jest umiejscowione oprogramowanie sieciowe. Prace w trybie sieciowym zapewniają moduły programowe, omówione poniżej.

● **Komputer obliczeniowy**. Głównym modułem oprogramowania jest moduł MŁOT (moduł łącznikowy obsługi transmisji), sterujący terminalami oraz tworzący połączenia sieciowe. Moduł komunikuje się z oprogramowaniem węzła w EC 8371.01 przez wydzielony podkanał multiplekserowy oraz z programem MCP realizującym metodę dostępu TCAM 5, która obsługuje terminale na poziomie fizycznym. Przepływ komunikatów między programem MŁOT a aplikacjami sieciowymi i terminalami zapewnia program MPC. Moduł MŁOT zakodowano w języku Pascal 360. Aplikacje sieciowe to TSO, SKOT i Stacja Transferu Zbiorów (protokół NIFTP z tzw. Blue Book).

● **Komputer czołowy i węzła.** Oprogramowanie składa się z dwóch zasadniczych części: programu sterującego obsługą terminali na poziomie fizycznym (NCP) oraz – programu WĘZEŁ. Program NCP współpracuje przez wydzielony podkanał z programem MCP, a program WĘZEŁ przez drugi podkanał z programem MŁOT. Program WĘZEŁ składa się ze stacji sieciowej metody dostępu, stacji transportowej, sieciowej i operatorskiej. Strukturę oprogramowania sieci przedstawiono na rys. 2.



Rys. 2. Struktura oprogramowania sieci SKJS2, wersja 1 (STZ – stacja transferu zbiorów)

Charakterystyka użytkowa sieci

Całość sieci składa się ze sprzętu dostarczanego przez ZE ELWRO we Wrocławiu. Poza modemami nie są wymagane dodatkowe urządzenia, co jest jedną z zalet sieci.

Podstawowymi zasobami sieciowymi są:
- System Kontroli i Obsługi Terminali (SKOT);
- podsystem wielodostępu (TSO);
- Stacja Transferu Zbiorów.

Inne aplikacje (np. pocztę elektroniczną) można utworzyć zgodnie ze specyficznymi potrzebami użytkownika lub w wyniku dalszego rozwoju sieci przez producenta. Dostęp do tych zasobów jest możliwy z każdego terminala sieci. Połączenie z wybraną aplikacją następuje po wydaniu z terminala polecenia NETON z odpowiednimi parametrami, a rozłączenie – po wydaniu polecenia NETOFF. Po rozłączeniu można połączyć się ponownie z dowolną aplikacją. Usługi komunikacyjne są realizowane przez oprogramowanie sieciowe i są niewidoczne dla użytkownika.

Operator konsoli operatorskiej, pełniącej jednocześnie funkcje konsoli sieciowej, ma możliwość prostego diagnozowania połączeń sieciowych, badania ich liczby i stanu, może też wymieniać informacje z operatorem innego komputera obliczeniowego sieci.

* * *

Całość podstawowego oprogramowania sieci przetestowano na konfiguracji złożonej z trzech zestawów maszynowych. Przy pracy terminalowej nie odczuwa się różnicy między współpracą z aplikacją lokalnego i zdalnego komputera obliczeniowego (praktycznie nie widać opóźnień czasowych).

Krótka charakterystyka pracy z aplikacjami sieciowymi

● **TSO** – wprowadzone zmiany umożliwiają pracę dupleksową, co wpływa na przyspieszenie wprowadzania danych z terminala; możliwe jest w każdej chwili wysłanie sygnału przerwania. Uzyskano też nowe możliwości pracy ze standardowym edytorem – po wyświetleniu na ekranie tekstu numerowanego można dokonać zmian we wszystkich wierszach i przesłać poprawiony tekst do edytora, przyciskając jeden klawisz.
● **SKOT** – pracuje tak samo jak dotychczas. Niedawno w ramach prac własnych Politechniki Wrocławskiej uruchomiono w sieci SKJS2, wersja 1, pod nadzorem systemu SKOT, system wyszukiwania danych bibliograficznych ISIS. Istnieje zatem możliwość korzystania z baz danych; baza używana na Politechnice Wrocławskiej zawiera wszystkie skatalogowane przez Bibliotekę Główną pozycje zwarte.
● **Stacja Transferu Zbiorów** jest jedyną aplikacją sieciową utworzoną od podstaw z myślą o pracy sieciowej. Umożliwia przesyłanie dowolnych zbiorów sekwencyjnych (po rozładowaniu można przesyłać zbiory biblioteczne i indeksowo-sekwencyjne).

LITERATURA

[1] Budzianowski L. i in.: Projekt techniczny oprogramowania PTD jako procesora czołowego i komputera węzła w sieci SKJS2, wersja 1. Instytut Cybernetyki Technicznej, Politechnika Wrocławska, 1984, 1985
[2] Dubielewicz A. i in.: Stacja Transferu Zbiorów w sieci SKJS2, wersja 1. Centrum Obliczeniowe Polltechniki Wrocławskiej, 1986
[3] Dubielewicz I. i in.: Aparatura pomiarowo-diagnostyczna dla sieci SKJS2, wersja 1. Centrum Obliczeniowe Politechniki Wrocławskiej, 1986
[4] Dyrka K. i in.: Zasady tworzenia i udostępniania zasobów w sieci SKJS2, wersja 1. Centrum Obliczeniowe Politechniki Wrocławskiej, 1986
[5] Huzar A., Jacukowicz M.: Terminal w sieci SKJS2, wersja 1. Centrum Obliczeniowe Politechniki Wrocławskiej, 1986
[6] Huzar A. i in.: Projekt techniczny oprogramowania EC 1032 jako komputera obliczeniowego w sieci SKJS2, wersja 1. Centrum Obliczeniowe Politechniki Wrocławskiej, 1986
[7] Kremienowski P. i in.: Dokumentacja eksploatacyjna zadań kontrolnych i pomiarowych do badań sieci SKJS2 wersja 1. IKSAiP, Wrocław, 1986.

Uprzejmie informujemy Czytelników, że egzemplarze INFORMATYKI – bieżące i archiwalne – można kupić nie tylko w kioskach Ruchu, Klubie NOT SIGMY, Zakładzie Kolportażu i Dziale Handlowym (szczegóły podano w WARUNKACH PRENUMERATY), ale również
w lokalu naszej redakcji
ul. Mickiewicza 18 m. 17 w Warszawie, tel. 39-14-34
oraz
w specjalistycznej księgarni PP „Domu Książki"
ul. Mokotowska 51/53 w Warszawie, tel. 28-16-14
Zapraszamy wszystkich zainteresowanych.

HALINA CIECHOMSKA
TERESA WÓJCIEKIAN
EWA ZABOROWSKA
Instytut Maszyn Matematycznych
Warszawa

PROMIS – System Zarządzania Bazą Danych (2)

W pierwszej części artykułu przedstawiono ogólną architekturę systemu PROMIS oraz omówiono takie jej elementy, jak baza danych, słownik danych, polecenia i programy usługowe dla Administratora Bazy Danych (ABD). Druga część zawiera omówienie pozostałych ważniejszych składników systemu.

PODSYSTEM PROMISCRIPT

PROMISCRIPT jest konwersacyjnym, wielodostępnym systemem umożliwiającym korzystanie z bazy danych pracującej pod kontrolą systemu PROMIS. Użytkownik podsystemu PROMISCRIPT ma do dyspozycji język PROMIS oraz język poleceń redakcyjnych.

Polecenia języka poleceń PROMIS są przeznaczone do wykonywania określonych operacji na bazie danych. Operacje te mogą dotyczyć albo całych rekordów, albo wystąpień (członów) grup krotnych. Polecenia redakcyjne umożliwiają użytkownikowi definiowanie sekwencji poleceń PROMIS i wykonywanie pewnych operacji na zdefiniowanej sekwencji.

Wszystkie polecenia redakcyjne oraz pojedyncze polecenia PROMIS, tzn. nie tworzące sekwencji, są wykonywane przez odpowiedni interpreter PROMISCRIPT bezpośrednio po wprowadzeniu.

Polecenia PROMIS

Stosując polecenia PROMIS użytkownik może wykonywać następujące operacje na bazie danych:
- otwierać i zamykać pliki (polecenia OPEN FILE i CLOSE FILE),
- wyszukiwać rekordy (polecenia FIND, FIND AND, FIND OR, START ISN, LOCATE KEY),
- kolejno sprawdzać wyszukiwane rekordy (polecenia GET DIRECT, GET NEXT),
- uzyskiwać informacje o liczbie rekordów spełniających dane kryterium (polecenia COUNT, COUNT AND, COUNT OR) oraz o liczbie rekordów, w których występuje dana wartość deskryptora (polecenie HISTOGRAM),
- formatować i wyświetlać wybrane rekordy (polecenie DISPLAY RECORD),
- modyfikować pojedynczą pozycję rekordu (polecenie CHANGE RECORD),
- usuwać rekord z bazy danych (polecenie PURGE RECORD),
- lokalizować i wyświetlać wystąpienie (człon) grupy krotnej (polecenie DISPLAY MEMBER),
- modyfikować pojedynczą pozycję wybranego wystąpienia grupy krotnej (polecenie CHANGE MEMBER),
- usuwać wystąpienie grupy krotnej (polecenie PURGE MEMBER);
- blokować i zwalniać rekordy (polecenia HOLD RECORD, RELEASE RECORD).

Ponadto użytkownik ma do dyspozycji polecenia służące do:
- wskazywania nazwy pliku, którego będą dotyczyć następne polecenia (polecenie FILE),
- wskazywania nazwy podschematu służącego do wyprowadzania danych (polecenie VIEW).

Polecenia PROMIS mogą być używane w postaci rozwiniętej lub krótkiej. Przykładem polecenia w postaci rozwiniętej może być zdanie („PRS〉" jest przynagleniem wysyłanym przez PROMISCRIPT):

```
PRS〉 10 FIND ALL THE RECORDS IN FILE:PERSON
PRS〉 WHEN SUMA = 〉 1000000 AND ROK = 〉 1982
PRS〉 BUT NOT 1984
```

Krótka postać tego samego polecenia jest następująca:

```
PRS〉 FI FILE:PERSON SUMA =〉 1000000 & ROK =〉 1982
% 1984
```

Krótka postać polecenia może zawierać odpowiedni skrót wybranej operacji PROMIS i znaki specjalne zastępujące operatory lub słowa kluczowe. Niektóre słowa kluczowe mogą być pominięte.

Polecenia redakcyjne

W celu zdefiniowania sekwencji poleceń należy poszczególne wiersze tej sekwencji (zawierające polecenia PROMIS) poprzedzić numerami określającymi kolejność wykonywania.

Przykładem sekwencji poleceń są zdania:

```
PRS〉 10 FIND ZAKLPR = IMM AND PLEC = M
PRS〉 20 COUNT AND STCYW = WOLNY
PRS〉 30 FIND AND RODZUBEZ = POLISA1
PRS〉 40 DISPLAY RECORD VIEW : PFULL
PRS〉 50 GET NEXT
```

Użytkownik ma do dyspozycji następujące polecenia redakcyjne:
LIST – wyświetlanie sekwencji poleceń lub jego fragmentu,
RUN – wykonywanie sekwencji poleceń lub jego fragmentu,
RESEQUENCE – porządkowanie sekwencji poleceń,
DELETE – usuwanie wiersza z sekwencji,
PRINT – wyprowadzenie na ekran tekstów komentarzy.
Na przykład polecenie:

```
PRS〉 RUN 10 – 30
```

spowoduje wykonanie wierszy oznaczonych numerami od 10 do 30, to znaczy pierwszych wierszy sekwencji podanej w poprzednim przykładzie, a polecenie:

```
PRS〉 RUN
```

spowoduje wykonanie całej tej sekwencji.

Parametryczna postać poleceń

PROMISCRIPT umożliwia takie konstruowanie poleceń, aby podczas ich wykonywania (często wielokrotnego) system prosił o wprowadzenie wartości wyróżnionych przez użytkownika parametrów. Wielkości sparametryzowane są umieszczane między znakami ⓐ . Na przykład, wprowadzenie polecenia:

```
PRS〉 10 FIND ⓐ NAZWA POZYCJI ⓐ = ⓐWARTOSCP ⓐ –
PRS〉 ⓐ WARTOSCK ⓐ % ⓐZ WYJĄTKIEM ⓐ
PRS〉 RUN
```

spowoduje następującą reakcję systemu (podkreślono wartości wprowadzane przez użytkownika):

ENTER NAZWA POZYCJI **DATUR**
ENTER WARTOSCP **1920**
ENTER WARTOSCK **1930**
ENTER Z WYJĄTKIEM **1927**
77 RECORDS FOUND
PRS〉

Wyszukiwanie rekordów

PROMISCRIPT umożliwia wyszukiwanie rekordów w następujący sposób:
- według podanego kryterium,
- według wewnętrznego identyfikatora ISN,
- kolejno, według podanego klucza (tzn. od wskazanej wartości deskryptora).

Wyszukiwanie rekordów według podanego kryterium jest realizowane przy użyciu poleceń FIND (skrót FI), FIND AND (skrót FIA) i FIND OR (skrót FIO). Polecenie FIND umożliwia użytkownikowi wybranie zbioru rekordów spełniających podane kryterium wyszukiwania. Efektem wykonania polecenia FIND jest:
- liczba rekordów spełniających dane kryterium (komunikowana użytkownikowi),

- lista wewnętrznych identyfikatorów ISN, wybranych rekordów, przechowywana przez system do dalszego przetwarzania.

Do wyszukiwania rekordów według wewnętrznego identyfikatora służą polecenia GET DIRECT i START ISN.

Polecenie GET DIRECT odczytuje z bazy danych rekord według podanego ISN i udostępnia go użytkownikowi zgodnie z podanym podschematem; na przykład:

PRS〉GD ISN = 10 VIEW:AFULL

Polecenie START ISN umożliwia użytkownikowi sekwencyjne przeglądanie bazy danych od podanego ISN. Polecenie to ustala pozycję początkową przeglądania. System nie korzysta przy tym z Asocjatora, tylko z Konwertera Adresów; na przykład:

PRS〉 SI 5 VIEW:PFULL

Do wyszukiwania rekordów kolejno według podanego klucza służy polecenie LOCATE KEY (skrót LK). Umożliwia ono użytkownikowi rozpoczęcie sekwencyjnego przeglądania rekordów według podanej wartości deskryptora; na przykład:

PRS〉 LK RODZUBEZ = POLISAZ

Po wykonaniu poprawnego polecenia wyszukiwania, o ile użytkownik podał nazwę podschematu określającego postać wyjściową rekordu, system wyświetla automatycznie pierwszy znaleziony rekord. Kolejne wyszukane rekordy można otrzymać podając polecenie GET NEXT (skrót GN). Odszukany rekord można również wyświetlić za pomocą polecenia DISPLAY RECORD (skrót DR) z odpowiednią nazwą podschematu.

Jeżeli użytkownika interesuje tylko liczba rekordów spełniających pewne kryterium, to może posłużyć się poleceniami COUNT (CO), COUNT AND (COA), COUNT OR (COO). Polecenie COUNT (CO) jest podobne do FIND; rezultatem jego wykonania jest liczba rekordów spełniających zadane kryterium. Po wykonaniu polecenia COUNT lista wewnętrznych identyfikatorów rekordów nie jest przechowywana przez system.

DOSTĘP Z PROGRAMÓW APLIKACYJNYCH

Programy użytkownika mogą współpracować z systemem PROMIS za pomocą odwołań CALL. PROMIS akceptuje takie odwołania, jeśli są zgodne ze standardem przyjętym w systemie DOC PB, np. odwołania CALL w językach Cobol, Fortran, Basic 2-Plus i Macro.

Każde odwołanie CALL powoduje wykonanie przez jądro systemu PROMIS określonego polecenia. W odwołaniu użytkownik powinien podać:
- kod polecenia,
- nazwę pliku,
- kryteria wyboru (jeśli polecenie dotyczy wyszukiwania rekordów),
- nazwę podschematu (jeśli polecenie dotyczy przesłania danych z (do) bazy.

Odpowiedź systemu może zawierać:
- informacje o ewentualnych błędach,
- liczbę znalezionych rekordów spełniających zadane kryterium,
- wewnętrzny identyfikator rekordu,
- obraz rekordu pobranego z bazy, odpowiadający zadanemu podschematowi.

Powyższe informacje są przekazywane w trzech obszarach komunikacyjnych zarezerwowanych w programie aplikacyjnym. Adresy tych obszarów stanowią parametry odwołań CALL. Pierwszy z obszarów (tzw. Blok Sterujący) jest przeznaczony na podstawowe informacje związane z wykonaniem polecenia (np. kod polecenia, nazwa pliku, kod powrotu), drugi obszar – na informację uzupełniającą, np. kryterium wyszukiwania, a trzeci – na przesyłany rekord.

Polecenia PROMIS wywoływane z programów użytkowych mogą obejmować wszystkie wcześniej wymienione podstawowe operacje na bazie danych oraz dodatkowe operacje pozwalające:
- dodawać rekordy do pliku (polecenie ADD RECORD),
- modyfikować cały rekord, a nie tylko pojedynczą pozycję (polecenie UPDATE RECORD),
- otwierać i zamykać kanały dostępu (polecenia OPEN THREAD i CLOSE THREAD).

GENERATOR SPRAWOZDAŃ PROMIWRITER

PROMIWRITER jest systemem umożliwiającym użytkownikowi bazy danych PROMIS generowanie sprawozdań z zawartości bazy przy użyciu prostych, niewielkich programów napisanych w specjalnym języku PRW. W programie PRW można zdefiniować następujące czynności:
- wyszukiwanie informacji z wielu plików bazy,
- uporządkowanie (sortowanie) otrzymanych informacji,
- wykonywanie operacji arytmetycznych,
- określenie formatu wyników ostatecznych,
- wskazanie wartości, które mają być przesłane na urządzenie wyjściowe.

Do zadań systemu PROMIWRITER związanych z przetwarzaniem programu PRW należy:
- określenie (przez dialog z użytkownikiem) urządzenia, na które ma być przesłane wygenerowane sprawozdanie,
- przetłumaczenie wskazanego programu PRW na język wewnętrzny,
- wykonanie przetłumaczonego programu PRW,
- przesłanie wyników programu na wskazane przez użytkownika urządzenie.

Danymi dla systemu PROMIWRITER są programy w języku PRW tworzone za pomocą edytora. W wyniku działania generatora sprawozdań otrzymuje się tekst programu oraz wyniki właściwe, tzn. odpowiednio sformatowane i przetworzone wartości z bazy. W wypadku błędnego programu kompilator PRW zaznacza, które fragmenty instrukcji są niezgodne ze składnią lub semantyką języka PRW. Wyniki działania programu można wydrukować, przesłać na dowolny spośród aktywnych monitorów ekranowych (niekoniecznie ten, z którego uruchomiono program) lub przechować w zbiorze dyskowym.

Język PRW

Język PRW jest językiem wysokiego poziomu służącym do przygotowywania programów dla generatora sprawozdań. Program PRW jest zbudowany z sekwencji instrukcji (ewentualnie uzupełnionych etykietami) i komentarzy. Instrukcje PRW działają na zmiennych, funkcjach systemowych, pozycjach z plików bazy, wskaźnikach daty i czasu, tekstach, liczbach, etykietach, wyrażeniach arytmetycznych, wyrażeniach logicznych.

Wartości pozycji z bazy są interpretowane przez program PRW jako wartości tekstowe lub liczbowe. Operacje arytmetyczne

mogą być wykonywane tylko na wartościach liczbowych. Ponadto obowiązuje zasada zgodności typów w relacjach występujących w wyrażeniach logicznych. Typ wartości pozycji w języku PRW jest określany na podstawie typu tej pozycji, zdefiniowanego w schemacie odpowiedniego pliku.

W języku PRW występują następujące rodzaje instrukcji:
- wyszukiwania z plików bazy,
- porządkowania otrzymanych wyników,
- formatowania postaci wyników końcowych,
- wyprowadzania,
- sterowania działaniem programu,
- pomocnicze (np. instrukcje wykonujące działania arytmetyczne).

Instrukcje wyszukiwania (BROWSE, FIND, HISTOGRAM, READ) działają podobnie do niektórych instrukcji dostępnych z programów aplikacyjnych i systemu PROMISCRIPT. Wszystkie instrukcje wyszukiwania umożliwiają odnajdywanie rekordów na podstawie wartości będących deskryptorami. Instrukcje FIND i READ pozwalają ponadto wyszukiwać na podstawie wartości innych pozycji.

Do porządkowania wartości wyszukanych z bazy służy instrukcja SORT. Danymi dla instrukcji SORT są rekordy wyszukane przez instrukcje BROWSE, FIND lub READ, ewentualnie tylko te z wyszukanych rekordów, które spełniły dodatkowe warunki nałożone instrukcjami ACCEPT i REJECT. Instrukcja SORT umożliwia sortowanie rekordów według wartości wielu pozycji i we wskazanym dla każdej pozycji porządku (malejącym lub rosnącym).

Na wartościach pozycji wyszukanych z bazy można wykonać działania arytmetyczne oraz obliczać wartości funkcji systemowych: MIN, MAX, AVER (średnia arytmetyczna), COUNT (liczba rekordów w podgrupie), OLD (wartość wskazanej pozycji z ostatniego rekordu w podgrupie), SUM (suma wartości pozycji w danej podgrupie), TOTAL (suma wartości pozycji ze wszystkich podgrup) oraz funkcji NMIN, NAVER i NCOUT – działających tylko dla wartości niezerowych (lub niepustych).

Do instrukcji sterujących działaniem programu PRW zalicza się instrukcję warunkową IF-THEN-ELSE, pętlę FOR-THRU-STEP i REPEAT-UNTIL, instrukcję złożoną DO-DOEND oraz instrukcje STOP i END zaznaczające logiczny i fizyczny koniec programu.

Program (zamieszczony na wydruku) wyszukuje w pliku AUTO 30 samochodów marki Jaguar, sortuje je według rosnących wartości pozycji ROKPROD i wypisuje w postaci zdefiniowanej instrukcją FORMAT. Wiersze programu poprzedzone gwiazdką są komentarzami objaśniającymi działanie instrukcji.

Program w języku PRW – wyszukiwanie według kryterium dotyczącego jednego pliku

```
*wypisz tytuł wyników,
WRITE TITLE 'SAMOCHODY MARKI JAGUAR'
*podaj hasło dostępu do pliku,
PASSW = AAA
*ogranicz liczbę wyników do 30,
LIMIT 30
*określ postać wyników:
*długość tekstów do 15 znaków,
*długość wiersza = 120 znaków,
*20 wierszy na stronie,
FORMAT AL = 15, LS = 120, PS = 20
*znajdź w pliku AUTO rekordy dotyczące samochodów marki
*Jaguar i przetwarzaj je według podschematu AAUTO,
FIND IN AUTO_AAUTO WHERE MARKA = 'JAGUAR'
*posortuj otrzymane rekordy rosnąco według roku produkcji,
SORT BY ROKPROD ASCENDING
*wypisz markę, rok produkcji, nr silnika i nr rejestracyjny,
*wyniki poprzedź numerem porządkowym; wszystkie kolumny,
*z wyjątkiem liczby porządkowej, mają mieć nagłówki standar-
*dowe tzn. takie jak nazwa danej pozycji,
DISPLAY 'Lp.' COUNTER MARKA ROKPROD NRSIL NRREJ
CLOSE LOOP
END
```

* * *

Na zakończenie artykułu, warto jeszcze podjąć próbę określenia miejsca systemu PROMIS wśród innych dostępnych systemów zarządzania bazą danych: czy baza danych PROMIS realizuje jakiś znany teoretyczny model danych i jakie są praktyczne możliwości systemu PROMIS, w porównaniu z możliwościami innych systemów baz danych, dostępnych na komputerach SM. Na pierwsze pytanie trzeba odpowiedzieć negatywnie, jakkolwiek baza danych PROMIS może przypominać bazę relacyjną, jeśli zrezygnuje się z definiowania pozycji grupowych i krotnych. PROMIS nie umożliwia jednak wykonywania typowych operacji relacyjnych (SELEKCJA, PROJEKCJA, ŁĄCZENIE).

Rozpatrując drugie zagadnienie trzeba zauważyć, że:

- Oprócz systemu PROMIS na komputerze SM-4 jest obecnie dostępny tylko system zarządzania bazą danych DTR (rozpowszechniony od 1984 roku przez FMiK Era), a system BAR jest w trakcie opracowywania (w IMM).
- W systemie DTR model danych jest zbliżony do modelu relacyjnego. Jest to system konwersacyjny; umożliwia on wykonywanie podstawowych operacji na bazie (wprowadzanie, modyfikowanie i wyszukiwanie danych), pod warunkiem posiadania przez użytkownika odpowiednich uprawnień. Do wykonywania tych operacji służy specjalny język o polskich słowach kluczowych, umożliwiających zagnieżdżanie poleceń i tworzenie procedur.
- System DTR jest oparty na systemie obsługi zbiorów dyskowych RCS, co umożliwia korzystanie z ogólnych programów usługowych, takich jak EDI lub PIP; programy te nie obejmują jednak wszystkich czynności potrzebnych do eksploatacji systemu zarządzania bazą danych. W porównaniu z systemem DTR system PROMIS ma następujące zalety:
  - wielodostęp,
  - możliwość dostępu do bazy z programów aplikacyjnych,
  - bogaty zestaw konwersacyjnych programów dla potrzeb Administratora Bazy Danych.
- System BAR będzie systemem zarządzania bazą danych opartym na modelu relacyjnym, z językiem dostępu SQL stosowanym w trybie wsadowym i konwersacyjnym. Prace nad systemem BAR nie zostały jeszcze zakończone, modelowa wersja systemu umożliwiająca wykonanie podstawowych operacji na bazie w sposób wsadowy, tj. z programów napisanych w języku Fortran lub Cobol i uzupełnionych zdaniami języka SQL, powinna być dostępna w końcu 1988 r., a jej rozszerzenia w tym również konwersacyjna wersja systemu – w terminie późniejszym.
- PROMIS miał do czerwca 1987 roku tylko dwa wdrożenia, ale jego pierwowzór ADABAS, działający na komputerach PDP-11, zyskał sobie dużą popularność i ma setki wdrożeń.

Obecnie do rozpowszechniania systemu PROMIS przystępuje FMiK Era.

Z przykrością informujemy naszych Czytelników, że począwszy od numeru lipcowego br. cena INFORMATYKI została ustalona na 250 zł za egzemplarz.

Nowa cena obowiązuje nowych prenumeratorów oraz sprzedaż odręczną. Jednak nie wyklucza się ewentualnej dopłaty różnicy (pomiędzy nową i starą ceną) do już opłaconej prenumeraty na drugie półrocze br. Prenumeratorzy zostaną zawiadomieni pisemnie przez Dział Kolportażu Wydawnictwa SIGMA.

Dotychczasowa cena ulgowa, 50 zł, zostaje utrzymana w prenumeracie do końca 1988 r.

JAN BIELECKI
Warszawa

Turbo C

Preprocesor i makrodefinicje

Czynności polegające na włączaniu do programu źródłowego tekstów zawartych w zbiorach danych, ustanawianiu i rozwijaniu makrodefinicji oraz włączaniu warunkowym są nazywane łącznie **przetwarzaniem wstępnym,** a program realizujący to przetwarzanie, najczęściej stanowiący integralną część kompilatora, jest nazwany **preprocesorem.** Dyrektywy preprocesora różnią się tym od innych napisów tworzących tekst źródłowy, że rozpoczynają się od znaku #, który[1] w wierszu, w którym wystąpił jest co najwyżej poprzedzony znakami spacji i tabulacji. Te dyrektywy, które nie mieszczą się w jednym wierszu, mogą być kontynuowane. Znakiem kontynuacji jest ukośnik umieszczony na końcu wiersza kontynuowanego. Zakresem i zasięgiem dyrektywy preprocesora jest moduł, w którym dyrektywa ta została użyta.

Przykład

```
#define max(a,b) a > b ?\
                 a : b
```

Przytoczona dyrektywa jest równoważna dyrektywie:

```
#define max(a,b) a > b ? a : b
```

i jest zapisana w dwóch wierszach, w których pierwszy jest kontynuowany.

Dyrektywy preprocesora dzielą się na cztery grupy. Do pierwszej należą dyrektywy włączające, do drugiej dyrektywy definiujące, do trzeciej dyrektywy włączania warunkowego, a do czwartej dyrektywy pomocnicze. W bieżącym odcinku omówiono dyrektywy dwóch pierwszych grup.

Dyrektywy włączające

Dyrektywy te mają postać:

```
#include "nazwa"
```

albo

```
# include <nazwa>
```

w której **nazwa** jest nazwą zbioru.

Zinterpretowanie dyrektywy # include powoduje włączenie w miejscu jej wystąpienia zawartości zbioru **nazwa.** Jeśli nazwa zbioru jest zawarta w nawiasach kątowych, to poszukiwanie zbioru ogranicza się do katalogu zbiorów włączanych, określonego za pomocą opcji kompilatora. Jeśli nazwa zbioru jest zawarta w cudzysłowach, to poszukiwanie w katalogu zbiorów włączanych jest poprzedzone poszukiwaniem zbioru w tym samym katalogu, z którego pochodzi zbiór zawierający dyrektywę # include. Oczywiście, jeśli zostanie podana kompletna nazwa zbioru, np. C:\JB-TC\JB-HEAD.H, to „poszukiwanie" jest zbyteczne i następuje natychmiastowe włączenie zawartości podanego zbioru. Pozostaje dodać, że nic nie stoi na przeszkodzie, aby zbiór włączany zawierał dyrektywy # include. W takim wypadku wymaga się jedynie, aby proces zagnieżdżonych włączeń był skończony.

[1] W technologii ISO znak ten nazywa się znakiem numeru (ang. number sign), po polsku – krzyżyk (przyp. red.).

Przykład

Jeśli zbiór EWA zawiera tekst:

```
main ()
#include <IZA>
```

zbiór IZA zawiera tekst:

```
{
#include "JAN"
```

a zbiór JAN zawiera tekst:

```
printf ("Hello, I am JanB");
```

to przekazanie preprocesorowi zbioru EWA spowoduje utworzenie modułu źródłowego:

```
main ()
{
  printf ("Hello, I am JanB");
}
```

Dyrektywy definiujące

Dyrektywy te mają postać:

#define **identyfikator ciąg-jednostek-leksykalnych**

albo postać:

#**define identyfikator lista-parametrów** \
ciąg-jednostek-leksykalnych

Zinterpretowanie pierwszej z wymienionych dyrektyw powoduje związanie z **identyfikatorem ciągu-jednostek-leksykalnych** i spowoduje, że każde wystąpienie **identyfikatora** będzie generować ten ciąg.

W drugiej z wymienionych dyrektyw **lista-parametrów** składa się z oddzielonych przecinkami identyfikatorów. Jeśli podczas interpretowania kolejnych wierszy tekstu źródłowego (nie zawierających dyrektyw # define) zostanie napotkany **identyfikator,** a po nim lista argumentów ujęta w nawiasy okrągłe, to cały taki napis, nazywany dalej makrowywołaniem, będzie generować napis utworzony z **ciągu-jednostek-leksykalnych,** w którym każdy z parametrów został zastąpiony odpowiadającym mu (być może pustym) argumentem.

Opisane czynności, nazywane rozwijaniem makrowywołań, są kontynuowane tak długo, jak długo może być wykonywane zastępowanie określone przez dyrektywy # define. Po ich zakończeniu następuje wygenerowanie otrzymanego napisu.

Jeśli w **ciągu-jednostek-leksykalnych** zawartych w makrodefinicji, występuje para sąsiadujących znaków #, być może otoczonych z każdej ze stron znakiem odstępu, to po dokonaniu wszystkich rozwinięć makrodefinicji, oba znaki #, wraz z otaczającymi je odstępami, zostaną usunięte. Właściwość ta jest najczęściej wykorzystywana do konstruowania nowych identyfikatorów.

Ponadto, jeśli pewien parametr makrodefinicji jest poprzedzony znakiem #, to podczas rozwijania makrodefinicji, argument odpowiadający temu parametrowi zostanie pozbawiony odstępów wiodących i kończących, a następnie otoczony z obu stron

znakiem " (cudzysłów). Jeśli tak przekształcony argument zawiera znaki " (cudzysłów) to każdy z nich zostanie poprzedzony ukośnikiem.

Makrodefinicje ustanowione za pomocą dyrektywy # define mogą być anulowane za pomocą dyrektyw o postaci:

#undef **identyfikator**

Anulowanie takie jest niezbędne, jeśli zamierza się ustanowić nową makrodefinicję, związaną z tym samym identyfikatorem, a **ciągi-jednostek-leksykalnych** w starej i nowej definicji nie są identyczne.

Przykłady

A. Użycie bazparametrowej dyrektywy # define

```
#define SIZE 200
int Arr[SIZE] [SIZE];
```

Zinterpretowanie przytoczonego napisu powoduje wygenerowanie napisu:

```
int Arr[200] [200];
```

B. Użycie dyrektywy # define z parametrami

```
#define max(a,b) a > b?a : b
main()
{

  int x = 5;
  printf("%d", max(x + 3,x - 2));
}
```

Zinterpretowanie przytoczonego napisu powoduje wygenerowanie napisu:

```
main()
{

  int x = 5;
  printf("%d", x + 3 > x - 2 ? x + 3 : x - 2);
}
```

C. Użycie operatora # undef

```
#define Kaja 13
out(Kaja)
#define Kaja 13
#undef Kaja
#define Kaja 14
out(Kaja)
```

Zinterpretowanie przytoczonego napisu powoduje utworzenie napisu:

```
out(13)
out(Kaja)
out(14)
```

Użycie dyrektywy # undef jest niezbędne, ponieważ jednostka leksykalna użyta w trzeciej dyrektywie # define jest różna od jednostki leksykalnej użytej w pierwszej i w drugiej z takich dyrektyw.

D. Ignorowanie makrowywołań w dyrektywach # define

```
#define Jan Ewa
#define Kaja Jan
#undef Jan
Kaja
```

Zinterpretowanie przytoczonego napisu powoduje wygenerowanie napisu:

Jan

a nie (jak można by przypuszczać) napisu:

Ewa

E. Ignorowanie makrowywołań w literałach znakowych i napisowych

```
#define j e
#define b we
#include <studio.h>
main()
{
  printf ("%c%s", 'j', "b");
}
```

Zinterpretowanie przytoczonego napisu powoduje wygenerowanie programu, którego wykonanie powoduje wyprowadzenie napisu jb, a nie (jak można by przypuszczać) napisu ewa.

F. Wielokrotne rozwijanie makrowywołań

```
#define alfa 5 + beta
#define beta gamma + 3
alfa - beta
```

Zinterpretowanie przytoczonego napisu powoduje wygenerowanie napisu:

```
5 + gamma + 3 - gamma + 3
```

G. Użycie operatora ##

```
#define Jan Ka
#define Ewa ja
#define Kaja Iza
#define Ka Ve
#define ja ra
#define Jan_Ewa Jan ## Ewa
```

Zinterpretowanie przytoczonego napisu powoduje utworzenie napisu:

Vera

a nie (jak można by przypuszczać) napisu Iza albo Jan_Ewa, choć **na skutek błędu implementacji, preprocesor języka Turbo C generuje ostatni z wymienionych napisów.**

H. Użycie operatora #

```
#define Show (var, fmt) printf(#var" ="fmt, var)
Show(One,"%d");
Show("Two","%s");
```

Zinterpretowanie przytoczonego napisu powoduje utworzenie napisu:

```
printf("One" ="%d", One);
printf("\"Two\"" = "%s", "Two");
```

Na skutek błędu implementacji, preprocesor języka Turbo C nie generuje ukośników.

I. Błędne użycie makrodefinicji

```
#define One fun(1),fun(2)
#define Two fun (3)
#define Max(x,y,z) x + y + z
Mac(One,Two)
```

Mimo iż napis One,Two generuje napis fun(1),fun(2),fun(3), który mógłby uchodzić za poprawną listę argumentów trzyparametrowej makrodefinicji Mac, wywołanie Mac(One,Two) zwiera jedynie dwa argumenty: fun(1),fun(2) i fun(3), a więc jest błędne. **Na skutek błędu implementacji, preprocesor języka Turbo C traktuje przytoczone makrowywołanie jako poprawne i generuje napis:**

```
fun(1) + fun(2) + fun(3)
```

LITERATURA

Bielecki J.: Język C interpretacja standardu. WNT 1988
Bielecki J.: Sam na sam z językiem C. WKiŁ 1988
Bielecki J.: Język C w systemie CP/M. WKiŁ 1988 (w druku)
Bielecki J.: Turbo C z grafiką dla IBM PC. WNT (w druku)
Borland International: Turbo C wersja 1.5. Borland 1988.

Jeszcze raz o Turbo Pascalu

Turbo Pascal firmy Borland International jest kompilatorem języka, o którym mówi się, że łączy zalety programowania strukturalnego w Pascalu, wygodę interakcyjnej pracy języka interpretowanego, takiego jak Basic, z szybkością języków kompilowanych. Niniejszy artykuł zawiera poglądowe omówienie tego języka i jego środowiska. Z uwagi na znaczną popularność Turbo Pascala w naszym kraju, przy braku podstawowych materiałów do jego nauki, takie syntetyczne wprowadzenie może być interesujące dla początkujących użytkowników.

Wykorzystanie języka kompilowanego, jakim jest Pascal, na komputerze osobistym zwykle oznacza ciągłe przechodzenie z edytora do kompilatora (i na odwrót), z częstą zmianą dyskietek. W związku z tym, firma Borland opracowała rozbudowany kompilator razem ze środowiskiem, w którym jest on używany. Wynikiem tego jest język wysokiego poziomu, który można stosować w poważnych projektach, ale na tyle łatwy i szybki, że można w nim pisać mniejsze programy aplikacyjne.

W artykule omówiono dwie wersje Turbo Pascala (dla systemów operacyjnych CP/M i PC-DOS), opisano ich cechy charakterystyczne i porównano z innymi profesjonalnymi kompilatorami Pascala.

ŚRODOWISKO

Po załadowaniu Turbo Pascala, na ekranie pojawia się menu główne (Turbo Main Menu). Z menu można wybrać pliki robocze i sposób, w jaki będą obsługiwane. Można też zobaczyć informacje na temat dostępnych zasobów. Każda pozycja menu jest opisana wyrazem lub zwrotem, a litera kluczowa jest podświetlona. Wpisanie pojedynczej (podświetlonej) litery powoduje wykonanie odpowiedniego polecenia z menu.

Z tego menu można redagować lub kompilować program, zmieniać współpracujące dyski, przeglądać skorowidze dysków lub zmieniać parametry kompilacji.

EDYTOR

Turbo Pascal jest wyposażony we wbudowany pełnoekranowy edytor przydatny przy nanoszeniu małych zmian w programie. Ma on polecenia przypominające zbiór poleceń znanego edytora Wordstar. Podczas instalowania może być dowolnie konfigurowany, na wzór innych edytorów tekstu, choć nie jest to łatwe zadanie i wymaga kilku prób.

Edytor jest bardzo wygodny. Błędy występujące podczas kompilowania programu powodują automatyczne przejście do trybu redagowania, a kursor wskazuje na linię programu, w której wystąpił błąd. Podczas wykonania programu skompilowanego, błędy wykonania, takie jak dzielenie przez zero, powodują również przejście do edytora. Dzięki temu, w porównaniu ze sposobami tworzenia programów w innych systemach, kolejne zmiany i kompilacje odbywają się bez większych kłopotów.

Edytor Turbo Pascala nie jest odpowiedni do redagowania dużych tekstów. Przy pewnych operacjach jego niedostatki są bardzo dokuczliwe i wtedy trzeba posługiwać się innym programem usługowym. Jednym z przykładów tych braków są polecenia zamiany ciągu: FIND i REPLACE. Mają one kilka opcji ograniczających lub rozszerzających zakres wykonywanego przeszukiwania i zamiany. Jednakże, każde nowe przeszukiwanie rozpoczyna się od początku. Oznacza to bardzo czasochłonną zamianę często powtarzających się ciągów.

Inną dziwną właściwością edytora jest definiowanie tabulatora. Większość edytorów ma ustalone lub definiowane znaczniki tabulacji. Użytkownicy są przyzwyczajeni do formatowania marginesu tekstu źródłowego (dosunięcia) za pomocą klawisza tabulacji. W edytorze Turbo Pascala znaczniki tabulacji są ustawiane od nowa w każdej linii. Tym samym tabulacja po powrocie karetki wyrównuje linię do początku poprzedniej. Taka metoda nie jest pozbawiona sensu, ale często zdarza się, że tabulacja nie działa tak, jak chciałby użytkownik. Niestety, nie ma możliwości użycia tradycyjnej metody tabulacji.

Edytor ma kilka innych, dyskusyjnych właściwości. Należy do nich programowe przełączanie trybów automatycznego wcinania (ang. auto-indent) lub wstawiania i zastępowania tekstu (ang. insert/overwrite), ale ich użycie jest tylko kwestią osobistego wyboru. Duża wygoda pracy z tym edytorem przeważa nad małymi błędami lub przeoczeniami projektowymi, które on zawiera.

KOMPILATOR

Program może być kompilowany i ładowany do pamięci w celu wykonania lub – zapisywany na dysk. Kompilacja z ładowaniem do pamięci jest szybka i pożyteczna przy uruchamianiu programów. Kompilowanie z zapisem na dysk może odbywać się dwoma sposobami. Można utworzyć plik wykonywalny (COM w systemie PC-DOS), który później inicjuje się z systemu operacyjnego. Można również tworzyć specjalne pliki łańcuchowe (typu CHN) wykorzystywane przez inne programy Turbo Pascala.

Kompilator jest szybki, ale nie jest dokładną implementacją definicji Pascala. Zaimplementowano w nim wszystkie podstawowe konstrukcje Pascala, ale istniejące różnice w domyślnej klauzuli instrukcji **case**, napisach, operacjach we-wy i stałych strukturalnych są znaczne.

STRUKTURY DANYCH

W Turbo Pascalu zaimplementowano wszystkie typy standardowe: całkowitoliczbowy, rzeczywisty, logiczny i znakowy. Nie istnieje długi typ całkowitoliczbowy i całkowitoliczbowy bezznakowy. Istnieją za to dwa inne typy proste: bajtowy, jako 8-bitowa liczba całkowita bez znaku, i napisowy, stanowiący jeszcze jedną implementację napisów w Pascalu. W deklaracji podaje się największą, przewidywaną długość napisu. Napis zadeklarowany jako **string** [80] może mieć dowolną długość nie przekraczającą 80 znaków. Deklaracja **string** [1] jest równoważna typowi standardowemu **char**.

Istnieje również cała gama procedur wbudowanych, służących do manipulowania napisami:
DELETE – usunięcie części napisu,
INSERT – wstawienie części napisu,
STR – konwersja typu całkowitoliczbowego na rzeczywisty,
VAL – konwersja typu napisowego na rzeczywisty lub całkowitoliczbowy,
COPY – kopiowanie części napisu,
CONCAT – konkatenacja dwóch napisów,

POS – znalezienie położenia wyszczególnionej części napisu. Dodatkowo, napisy można dodawać za pomocą operatora „+", co jest równoważne procedurze CONCAT.

Tak jak w innych wersjach Pascala, w Turbo Pascalu można stosować typy wyliczeniowe. Różnica dotyczy tylko operacji we-wy. Według definicji Pascala program przedstawiony na wydruku 1 powoduje wyprowadzenie na urządzenie wyjściowe napisu "BETA". W Turbo Pascalu wystąpi błąd kompilacji.

```
program test(output);
var
  foo: (alpha,beta,gamma);

begin
foo := beta;
writeln(output,foo);
end.
```

Wydruk 1

Pewną nowością są stałe o określonym typie. Jest to sposób tworzenia stałych dowolnego, poprzednio zdefiniowanego typu, np. tablicowego, napisowego, rekordowego, a nawet rekordowego z wariantami. Takie możliwości mają często bardziej zaawansowane języki programowania. Użycie stałych o określonym typie zilustrowano na wydruku 2. W ten sposób SpecialCommands staje się zmienną tylko odczytywalną, będącą typu CommandTable. Ponieważ można tak postępować z dowolnymi typami, uzyskuje się wiele sposobów na zaoszczędzenie czasu i miejsca w pamięci przy inicjalizacji struktur danych.

```
type
   CommandTable = array[1..5] of string[8];
const
   SpecialCommands: CommandTable =
     ('RUN','STOP','LIST','COMPILE','QUIT');
```

Wydruk 2

Turbo Pascal jest dostarczany z pierwotnie zdefiniowanymi strukturami danych odwzorowującymi pamięć i porty we-wy, można więc wykonywać operacje równoważne instrukcjom Basika – PEEK i POKE.

OPEROWANIE PLIKAMI

Do pliku odwołuje się przez zmienną plikową. Pierwszą operacją wykonaną na zmiennej plikowej musi być ASSIGN, czyli przypisanie nazwy spełniającej reguły DOS-a do zmiennej plikowej. Inne operacje wykonywane tylko na zmiennych plikowych są następujące:
REWRITE – otwarcie pliku do zapisu,
RESET – otwarcie pliku do odczytu,
READ – odczytanie elementu pliku,
WRITE – zapisanie elementu do pliku,
SEEK – przejście do określonego miejsca pliku
FLUSH – uaktualnienie pliku,
CLOSE – zamknięcie pliku (koniec operacji wykonywanych na nim),
ERASE – usunięcie pliku z dysku,
RENAME – nadanie plikowi innej nazwy,
EOF – wykrycie końca pliku,
FILEPOS – podanie pozycji elementu w pliku,
FILESIZE – podanie wielkości pliku.

Dodatkowo są dostępne zwykłe procedury dla plików tekstowych, takie jak READLN, WRITELN lub EOLN. Zwraca uwagę nieobecność w Turbo Pascalu funkcji GET i PUT. Zostały one słusznie wyeliminowane, ponieważ żadna inna konstrukcja języka (z wyjątkiem wskaźników) nie jest tak niejasna dydaktycznie. Procedury READ i WRITE, zawarte w Turbo Pascalu, w sposób zadowalający rozwiązują problemy, do których zaprojektowano GET i PUT.

INSTRUKCJA CASE

Jak już wspomniano, domyślna klauzula instrukcji **case** (tj. ewentualność wykonywana wtedy, gdy selektor nie pasuje do żadnej innej zdefiniowanej ewentualności) jest bardzo pożyteczną właściwością języka. Wydaje się, że sposobów implementacji klauzuli domyślnej jest tyle, ile implementacji Pascala. Firma Borland pokazała jeszcze jeden – chyba nie najszczęśliwszy. Przykład instrukcji **case** wzięty z firmowego podręcznika przedstawiono na wydruku 3.

```
case Year of
   Min..1939: begin
                Time := PreWorldWar2;
                write('Na świecie panował pokój..');
              end;
   1946..Max; begin
                Time := PostWorldWar2;
                write('Budowa nowego świata.');
              end;
else
  Time := WorldWar2;
  write('Stan wojny');
end;
```

Wydruk 3

Można zauważyć, że instrukcje występujące po **else** nie są ujęte w nawiasy syntaktyczne **begin** i **end**. Zwraca uwagę również to, że średnik poprzedza **else**. Te nieregularności nie mają precedensu w reszcie gramatyki Pascala. Wydaje się, że implementatorzy próbowali stworzyć instrukcję **case..else** przypominającą **if..then..else**. Niestety, jedyne co udało się osiągnąć to instrukcja niedopasowana do gramatycznych konwencji Pascala. Problem polega więc na tym, że **else** nie jest złe z powodu niepasowania do standardu, ale dlatego, że gramatyka **case..else** jest niezgodna z charakterem Pascala.

Stałe o określonym typie wprowadzone w Turbo Pascalu nie są zgodne ze standardem, ale wynikają ze sposobu definiowania typów, zmiennych i stałych. Jest to naruszenie tylko standardu, a nie ducha języka. Sposób zdefiniowania klauzuli **else** dla instrukcji **case** jest zaś naruszeniem zarówno ducha, jak i standardu Pascala.

KONTROLA TYPÓW

Pascal uważa się za język o stałej ścisłej typizacji, co oznacza, że kompilator próbuje przede wszystkim rozstrzygnąć, czy użytkownik zadeklarował wszystkie zmienne i parametry, a później egzekwuje te decyzje w całym programie. Jeśli zadeklaruje się zmienną typu REAL, to nie można jej przypisać wartości typu INTEGER.

Jednakże, niekiedy zachodzi potrzeba konwersji typów i dlatego w standardowym Pascalu zawarto kilka funkcji konwersji z nadzieją, że wyczerpują wszystkie możliwe przypadki. Przykładami mogą być funkcje FLOAT zamiany typu INTEGER na REAL i TRUNC zamiany typu REAL na INTEGER.

W standardowym Pascalu nie można wykorzystać luk w definicji języka i napisać własnych funkcji FLOAT i TRUNC. Są to zawsze procedury przygotowane przez autorów kompilatora. W Turbo Pascalu, natomiast można napisać własne procedury konwersji. Kompilator zawiera zestaw rozszerzeń Pascala umożliwiających unikanie kontroli typów. Pozwala to robić konsekwentnie wiele rzeczy, które programiści Pascala wykonywali dotąd za pomocą sztuczek.

Załóżmy, że trzeba napisać procedurę konkatenacji dwóch napisów. Procedura powinna działać dla dwóch napisów typu **string**[5] i **string**[80]. W standardowym Pascalu należałoby napisać oddzielną procedurę dostosowaną do długości napisu. W Turbo Pascalu natomiast, można łatwo napisać jedną procedurę obsługującą wszystkie przypadki.

Taki sposób programowania kryje w sobie pewne niebezpieczeństwa. Nie istnieje możliwość ograniczenia niezależności typów do pewnego ich zbioru. Tak więc, po przesłaniu niewłaściwych parametrów do procedury o niezależności typów, mogą zdarzyć się nie przewidziane sytuacje. Zwolnienie z kontroli typów nie jest jednak w Turbo Pascalu domyślne, więc przypadkowe błędy zostaną prawdopodobnie wykryte.

Tak się składa, że pisanie procedur konwersji w Turbo Pascalu jest nie tylko możliwe, ale w wypadku procedury FLOAT

konieczne. Standardowy kompilator jest bowiem wyposażony w procedurę TRUNC, a nie FLOAT. Tym samym nie ma wbudowanych możliwości zamiany typu całkowitoliczbowego na rzeczywisty.

PROCEDURY ZEWNĘTRZNE

Turbo Pascalowi brakuje jednej ważnej właściwości – możliwości tworzenia biblioteki procedur zewnętrznych. Możliwość rozłącznej kompilacji pozwalałaby utworzyć zbiór procedur i funkcji, których kod byłby współdzielony przez wiele programów. Dodatkowo, rozłączna kompilacja i testowanie części programu pomagałoby przyspieszyć cykl tworzenia nowego oprogramowania.

Ponieważ kompilator tłumaczy program źródłowy w Turbo Pascalu bezpośrednio na pliki wykonywalne i nie wykonuje się zewnętrznej konsolidacji, to nie można oddzielnie kompilować modułów jednego programu aplikacyjnego. Firma Borland poświęciła rozłączną kompilację na rzecz szybkości tworzenia programu. Taka decyzja na pewno nie cieszy twórców dużych programów.

Istnieje prosty sposób wywoływania procedur zewnętrznych, pod warunkiem znajomości ich adresów, ale nie rozwiązuje to problemów wcześniej poruszonych.

W Turbo Pascalu można wywoływać inne programy za pomocą procedur EXECUTE i CHAIN. Procedura EXECUTE przesyła sterowanie do dowolnego programu, który można wykonywać na żądanie systemu operacyjnego. Gdy program kończy się, sterowanie wraca do systemu operacyjnego, a nie do programu wykonującego EXECUTE. Procedura CHAIN przekazuje sterowanie do programu skompilowanego w specjalny sposób. Programy kompilowane normalnie zawierają wbudowaną bibliotekę procedur standardowych. Przy kompilacji na plik typu CHN (tzw. łańcuchowy) biblioteka nie jest dołączana i plik dyskowy jest dużo mniejszy. Przy ładowaniu nowego programu za pomocą wywołania CHAIN, biblioteka dotychczas rezydująca w pamięci jest zatrzymywana, ale sam program aplikacyjny jest zamieniany. Po zakończeniu programu sterowanie wraca do systemu operacyjnego.

Podczas kompilacji można również dołączać zewnętrzne pliki źródłowe. Tym samym dwa programy mogą współdzielić kod źródłowy, nie współdzieląc kodu skompilowanego.

INNE WŁAŚCIWOŚCI

Turbo Pascal umożliwia wplatanie kodu maszynowego (ang. in-line), choć w bardzo prymitywny sposób. Instrukcja **inline** akceptuje ciągi liczb i umieszcza je bezpośrednio w programie kompilowanym. Za jej pomocą można pisać procedury obsługi przerwań lub optymalizować szybkość wybranych procedur. Jest to cecha niestandardowa, ale bardzo pożyteczna.

Pakiet firmy Borland umożliwia bezpośrednie wywołanie procedur systemu operacyjnego. Pozwala to pisać programy zmieniające środowisko tworzone przez system operacyjny. Dobrym przykładem takiego programu jest drukowanie skorowidza dysku.

* * *

Porównania Turbo Pascala z innymi kompilatorami są trochę nie na miejscu. Z pewnością Turbo Pascal nie jest kompilatorem zabawkowym, do rozwiązywania problemów mniejszej wagi.

```
{ Program wyznaczania liczb pierwszych metodą}
{ sita Eratostenesa }
program ERATOSTENES;
const
  size = 8190;
var
  flags : array [0..size] of boolean;
  i, prime, k, count, iter : integer;

begin
writeln('10 iteracji');
for iter := 1 to 10 do
  begin
  count := 0;
  for i := 0 to size do
    flags[i] := true;
  for i := 0 to size do
    if flags[i] then
      begin
      prime := i + i + 3;
      (* writeln(prime); *)
      k := i + prime;
      while k <= size do
        begin
        flags[k] := false;
        k := k + prime;
        end;
      count := count + 1;
      end;
  end;

writeln(count, ' liczb pierwszych');
end.
```

Wydruk 4

```
program PUZZLE;
{ Program rozwiązuje prostą łamigłówkę }
const
  size     = 511;
  classMax = 3;
  typeMax  = 12;
  d        = 8;
type
  pieceClass = 0..classMax;
  pieceType  = 0..typeMax;
  position   = 0..size;
var
  pieceCount : array [pieceClass] of 0..13;
  class      : array [pieceType] of pieceClass;
  pieceMax   : array [pieceType] of position;
  puzzle     : array [position] of boolean;
  p          : array [pieceType, position] of boolean;
  m, n       : position;
  i, j, k    : 0..13;
  kount      : integer;

function fit (i: pieceType; j: position): boolean;
label 1;
var
  k: position;
begin fit := false; for k := 0 to pieceMax[i] do
  if p[i,k] then if puzzle[j + k] then goto 1;
fit := true;
1:
end;

function place (i: piecetype; j: position): position;
label 1;
var
  k: position;

begin
  for k := 0 to pieceMax[i] do
    if p[i,k] then puzzle[j + k] := true;
  pieceCount[class[i]] := pieceCount[class[i]] - 1;
  for k := j to size do
    if not puzzle[k] then begin
      place := k;
      goto 1;
    end;
  writeln('Łamigłówka wypełniona');
  place := 0;
1:
end;

procedure remove (i: pieceType; j: position);
var k: position;
begin
  for k := 0 to pieceMax[i] do
    if p[i,k] then puzzle[j + k] := false;
  pieceCount[class[i]] := pieceCount[class[i]] + 1;
end;

function trial (j: position): boolean;
label 1;
var
  i : piecetype;
  k : position;
begin
  for i := 0 to typeMax do
    if pieceCount[class[i]] <> 0 then
      if fit (i, j) then begin
        k := place (i, j);
        if trial(k) or (k = 0) then begin
          writeln('Piece',i + 1,'at ',k + 1)
          trial := true;
          goto 1;
        end else remove (i, j);
      end;
  trial := false;
1: kount := kount + 1;
end;
begin
  writeln('Rozpoczęcie Łamigłówki');
  for m := 0 to size do puzzle[m] := true;
  for i := 1 to 5 do for j := 1 to 5 do for k := 1 to 5 do
    puzzle [i + d*(j + d*k)] := false;
  for i := 0 to typeMax do for m := 0 to size do p[i, m] := false;
  for i := 0 to 3 do for j := 0 to i do for k := 0 to 0 do
    p[0, i + d*(j + d*k)] := true;
  class[0] := 0;
  pieceMax[0] := 3 + d*1 + d*d*0;
  for i := 0 to 1 do for j := 0 to 0 do for k := 0 to 3 do
    p[1, i + d*(j + d*k)] := true;
  class[1] := 0;
  pieceMax[1] := 1 + d* + d*d*3;
  for i := 0 to 0 do for j := 0 to 3 do for k := 0 to i do
    p[2,i +d*(j + d*k)] := true;
  class[2] := 0;
  pieceMax[2] := 0 + d*3 + d*d*1;
  for i := 0 to 1 do for j := 0 to 3 do for k := 0 to 0 do
    p[3, i + d*(j + d*k)] := true;
  class[3] := 0;
  pieceMax[3] := i + d*3 + d*d*0;
  for i := 0 to 3 do for j := 0 to 0 do for k := 0 to 1 do
    p[4, i + d*(j + d*k)] := true;
```

```
class[4] := 0;
pieceMax[4] := 3 + d*0 + d*d*1;
for i := 0 to 0 do for j := 0 to 1 do for k := 0 to 3 do
  p[5, i + d*(j + d*k)] := true;
class[5] := 0;
pieceMax[5] := 0 + d*1 + d*d*3;
for i := 0 to 2 do for j := 0 to 0 do for k := 0 to 0 do
  p[6, i + d*(j + d*k)] := true;
class[6] := 1;
pieceMax[6] := 2 + d*0 + d*d*0;
for i := 0 to 0 do for j := 0 to 2 do for k := 0 to 0 do
  p[7, i + d*(j + d*k)] := true;
class[7] := 1;
pieceMax[7] := 0 + d*2 + d*d*0;
for i := 0 to 0 do for j := 0 to 0 do for k := 0 to 2 do
  p[8, i + d*(j + d*k)] := true;
class[8] := 1;
pieceMax[8] := 0 + d*0 + d*d*2;
for i := 0 to 1 do for j := 0 to 1 do for k := 0 to 0 do
  p[9, i + d*(j + d*k)] := true;
class[9] := 2;
pieceMax[9] := 1 + d*1 + d*d*0;
for i := 0 to 1 do for j := 0 to 0 do for k := 0 to 1 do
  p[10, i + d*(j + d*k)] := true;
class[10] := 2;
pieceMax[10] := 1 + d*0 + d*d*1;
for i := 0 to 0 do for j := 0 to 1 do for k := 0 to 1 do
  p[11, i + d*(j + d*k)] := true;
class[11] := 2;
pieceMax[11] := 0 + d*1 + d*d*1;
for i := 0 to 1 do for j := 0 to 1 do for k := 0 to 1 do
  p[12, i + d*(j + d*k)] := true;
class[12] := 3;
pieceMax[12] := 1 + d*1 + d*d*1;
pieceCount[0] := 13;
pieceCount[1] := 3;
pieceCount[2] := 1;
pieceCount[3] := 1;
m := 1 + d*(1 + d*1);
kount := 0;
if fit(0, m) then n := place(0, m) else writeln('Błąd');
if trial(n) then writeln('Sukces w', kount,'próbach')
else writeln('Niepowodzenie');
end.
```

Wydruk 5

Tabela 1. Wyniki testów dla kompilatorów pracujących pod kontrolą systemu operacyjnego PC-DOS 2.0

Czas wykonania (s)	Sito	Łamigłówka	Szybkie sortowanie
Turbo Pascal:			
Kompilacja z załadowaniem do pamięci	0,8	5,7	3,4
Kompilacja na plik COM	2,0	3,6	4,0
Wykonanie w pamięci	15,0	69,0	1,3
Wykonanie pliku COM	15,0	69,0	1,1
Pascal IBM:			
Wykonanie pliku EXE	76,0	416,0	1,2
Wielkość plików (KB)			
Plik COM (Turbo Pascal)	9	13	11
Plik EXE (Pascal IBM)	33	35	24

Tabela 2. Wyniki testów dla kompilatora pracującego pod kontrolą systemu operacyjnego CP/M 2.2

Czas wykonania (s)	Sito	Łamigłówka	Szybkie sortowanie
Kompilacja z załedowaniem do pamięci	0,5	3,0	2,0
Kompilacja na plik COM	5,0	12,0	9,0
Wykonanie w pamięci	23,0	77,0	1,0
Wykonanie pliku COM	25,0	77,0	1,0
Wielkość pliku COM KB	8	12	10

Nie nadaje się również do tworzenia dużego oprogramowania aplikacyjnego. Tak naprawdę, trzeba by go porównać z Basikiem. Przy posiadaniu Turbo Pascala praktycznie znika potrzeba korzystania z Basika. Kompilator firmy Borland jest dobrze napisany, przyjemny w użyciu i na tyle szybki, aby przesłonić pewne swoje wady. Można go również wykorzystywać do tworzenia kodu, który będzie tłumaczony przez kompilatory profesjonalne.

Wydruki 4, 5 i 6 zawierają znane programy testowe, wykorzystane do porównania Turbo Pascala z Pascalem IBM. Testy zostały dobrane nie z myślą o dokładnym zmierzeniu różnic, ale żeby dać ogólny obraz efektywności kompilatorów.

W tabelach 1 i 2 przedstawiono wyniki testów. Czas kompilacji liczono od wydania polecenia COMPILE do ponownego wyświetlenia znaku gotowości. Tym samym, przy kompilacji na wykonywalne pliki COM bardzo ważnym elementem jest szybkość działania napędów dyskowych. Testy wykonano na mikrokomputerze IBM PC z pamięcią 320 KB i dwoma napędami dysków dwustronnych o podwójnej gęstości. Czas wykonywania programu na pliku COM mierzono od początkowego komunikatu, obliczając czas załadowania i rozpoczęcia programu przez system operacyjny.

```
program QuickSort(input,output);
(* Program sortuje tablicę liczbową za pomocą algorytmu
   szybkiego sortowania  adaptowanego z trzeciego tomu
   Art of Computer Programming D. Knutha )
( W celu użycia algorytmu w teście wprowadzono następujące
   poprawki:
-  Liczby sortowane są typu rzeczywistego, co jest przypadkiem
   trudniejszym od 16-bitowych całkowitych
-  Tablica wejściowa jest wpisana w tekście.
-  W celu dokładniejszego zmierzenia czasu wykonania
-  ciało procedury printArray zawarto w komentarzach )
const
  max = 100;                ( najwiekszy rozmiar tablicy)
type
  standardArray = array [0..max] of real;
var
  numbers: standardarray;  ( tablica zawierajaca liczby do sortowania)
  last: integer;

procedure swap(var a,b: real);
  ( zamiana wartosci A i B)
var
  t: real;                  ( zmienna tymczasowa )

begin
  t := a;
  a := b;
  b := t;
end;
procedure printArray(top: integer);
  (wyswietla biezaca zawartosc tablicy)
  ( Można wpisać własną procedurę drukowania )
var
  i : integer;
begin
  for i := 0 to top         ( podczas testu jest to komentarz )
  do
  write(numbers[i]:8:2);
  writeln;
  writeln('-MARK-');       ( wykorzystawany podczas testow )
end;

procedure getArray(var top: integer);
( Wypełnia tablicę na podstawie danych wejściowych )
( W praktycznie wykorzystywanym programie dane mogą być pobierane
   ze standardowego urządzenia wejściowego lub pliku. Na potrzeby
   testu zainicjalizowano tablicę w odwrotnym porządku, jest to
   najtrudniejszy przypadek sortowania. )
const
  worstCase: standardArray =

  (100.0,99.0,98.0,97.0,96.0,95.0,94.0,93.0,92.0,91.0,
   90.0,89.0,88.0,87.0,86.0,85.0,84.0,83.0,82.0,81.0,
   80.0,79.0,78.0,77.0,76.0,75.0,74.0,73.0,72.0,71.0,
   70.0,69.0,68.0,67.0,66.0,65.0,64.0,63.0,62.0,61.0,
   60.0,59.0,58.0,57.0,56.0,55.0,54.0,53.0,52.0,51.0,
   50.0,49.0,48.0,47.0,46.0,45.0,44.0,43.0,42.0,41.0,
   40.0,39.0,38.0,37.0,36.0,35.0,34.0,33.0,32.0,31.0,
   30.0,29.0,28.0,27.0,26.0,25.0,24.0,23.0,22.0,21.0,
   20.0,19.0,18.0,17.0,16.0,15.0,14.0,13.0,12.0,11.0,
   10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0,
    0.0);

begin                                        ( getArray )
  top := 100;
  numbers := worstCase;
  printArray(top);
end;  ( getArray )

procedure bubbleSort (start, top: integer; var subArray: standardArray);
  ( procedura sortowania pecherzykowego )

var
  index: integer;
  switched: boolean;

begin repeat
    begin
    switched := false;
    for index := start to top - 1 do
    begin
      if subArray[index] > subArray[index + 1] then
      begin
        swap(subArray[index],subArray[index + 1]);
        switched := true;
      end;
    end;
    end
  until switched = false;
end;

procedure findMedian(start, top: integer; var subArray: standardArray);
  ( Znajdowanie wartosci medialnej i umieszczenie jej na poczatku
     sortowanej czesci tablicy )

var
  middle: integer;
  sorted: standardArray;

begin                                        ( findMedian )
  middle := (start + top)div 2;
  sorted[1] := subArray[start];
```

```
  sorted[2] := subArray[top];
  sorted[3] := subArray[middle];
  bubbleSort(1,3,sorted);
  if sorted[2] = subArray[middle] then
    swap(subArray[start],subArray[middle])
  else
    if sorted[2] = subArray[top] then
      swap(subArray[start],subArray[top]);
end;                                        { findMedian }

procedure sortSection(start, top: integer);
  { sortowanie czesci glownej tablicy i dzielenie na dwie
    czesci do sortowania }

var
  swapUp: boolean;
  s, e, m: integer;

begin                                       { sortSection }
  if top - start < 6 then                   { male czesci sortowane pecherzykowo
    bubbleSort(start,top, numbers)
  else
  begin
    findMedian(start, top, numbers);
    swapUp := true;

    s := start;
    e := top;
    m := start;
    while e > s do
    begin
      if swapUp = true then
      begin
        while (numbers[s] >= numbers[m]) and (e > m) do
          e := e -1;
        if e > m then
        begin
          swap(numbers[e], numbers[m]);
          m := e;
        end;
        swapUp := false;
      end
      else
      begin
        while (numbers[s] <= numbers[m]) and (s < m) do
          s := s + 1;
        if s < m then
        begin
          swap(numbers[s], numbers[m]);
          m := s
        end;

        swapUp := true;
      end;
    end;
    sortSection(start, m - 1);
    sortSection(m + 1, top);
  end;
end;                                        { sortSection }
begin                                       { program glowny }
  getArray(last);
  sortSection(0, last);
  printArray(last);
end.
```

Wydruk 6

Wszystkie pomiary wykonano co najmniej trzy razy, a wyniki w tabelach są średnią. Wszystkie czasy dłuższe od 2 s podano z dokładnością do 0,5 s.

Opracował: MARIUSZ KUC
na podstawie BYTE, July 1984

Konferencje

MASAD '88

Instytut Konstrukcji i Eksploatacji Maszyn Politechniki Wrocławskiej pod auspicjami Komitetu Budowy Maszyn Polskiej Akademii Nauk, przy udziale Zakładów Elektronicznych ELWRO oraz Klubu Użytkowników Mikrokomputerów profesjonalnych NOT organizuje **we Wrocławiu w dniach 28 listopada – 1 grudnia 1988 r.** międzynarodową konferencję naukową pn. **„Zastosowanie mikrokomputerów w analizie wytrzymałościowej i projektowaniu konstrukcji – MASAD'88".**

Celem konferencji jest przedstawienie bieżących kierunków zastosowań mikrokomputerów w analizie wytrzymałościowej i projektowaniu konstrukcji w Polsce na tle tendencji światowych oraz sformułowanie prognoz i zaleceń na temat rozwoju tej problematyki na najbliższe lata.

Problematykę konferencji stanowić będą podstawy naukowo-metodologiczne, przykłady aplikacyjne (w tym oprogramowanie) i zagadnienia zastosowania mikrokomputerów w następujących obszarach związanych z tematem konferencji:

- projektowanie konstrukcji (metodyka, optymalizacja, grafika itp.),
- analiza wytrzymałościowa konstrukcji,
- stanowiska pracy konstruktora (urządzenia, programy, obsługa),
- wspomaganie eksperymentu,
- techniki symulacyjne,
- badania eksploatacyjne konstrukcji.

Poszczególne sesje konferencji będą odbywać się w Gmachu Głównym Politechniki Wrocławskiej. Zakwaterowanie uczestników konferencji przewiduje się w hotelach we Wrocławiu.

Całodzienne wyżywienie zorganizowane zostanie w stołówce Politechniki Wrocławskiej.

Oficjalne języki konferencji:
– polski,
– angielski,
– rosyjski.

Koszt uczestnictwa 15 tys. zł od osoby.

Zgłoszenie uczestnictwa prosimy przesłać pod adresem sekretariatu konferencji.

Adres sekretariatu:
MASAD '88
J. A. Samołyk
Instytut Konstrukcji i Eksploatacji Maszyn
Politechniki Wrocławskiej
50–370 WROCŁAW, ul. I. Łukasiewicza 7/9
tel.: 48-42-74(76), teleks: 712254 PWR PL

Warunki prenumeraty na 1988 rok

Prenumeratorzy zbiorowi – jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat wyłącznie na blankiecie „wpłata-zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia).

Blankiety te będą dostarczane dotychczasowym prenumeratom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemne lub telefoniczne) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni – osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: NBP III Oddział Warszawa 1036-7490-139-11.

Prenumerata ulgowa – przysługuje wyłącznie osobom fizycznym – członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po jednym egzemplarzu każdego czasopisma.

Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumerata ze zleceniem wysyłki za granicę – zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy.

Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach:

– do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny,
– do 28 lutego na II, III i IV kwartał oraz II półrocze,
– do 31 maja na III i IV kwartał oraz II półrocze,
– do 31 sierpnia na IV kwartał.

Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela – Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00–716 Warszawa), skr. poczt. 1004, 00–950 Warszawa, tel. 40-00-21 wew. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism – można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 27-43-65), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00–950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena egzemplarza 250 zł (począwszy od nr 7, 1988), cena ulgowa 50 zł.

CENA		Prenumerata normalna		Prenumerata ulgowa (bez zmiany)	
normalna	ulgowa (bez zmiany)	kwartalna	półroczna	kwartalna	półroczna
250 zł	50 zł	750 zł	1500 zł	150 zł	300 zł

W pogoni za efektywnością komputerów

Sztuczna inteligencja, przetwarzanie obrazów są dziedzinami, w których badania są prowadzone od wielu lat. Nowym zjawiskiem jest natomiast możliwość technicznego rozwiązania tych problemów. Jeszcze do niedawna było to możliwe tylko przy wykorzystaniu dużych komputerów, obecnie coraz wydajniejsze są realizacje na stacjach roboczych (SUN, Xerox), a także na niektórych mikrokomputerach (IBM PC, Macintosh). Firma Neutron Data stworzyła dla mikrokomputera Macintosh oprogramowanie systemów ekspertowych, które zdaniem specjalistów okazało się lepsze niż oczekiwano.

W tradycyjnych architekturach komputerów całe oprogramowanie – zarówno system operacyjny jak i oprogramowanie specjalistyczne (np. graficzne) – było tworzone tylko dla jednego procesora. Z czasem tę architekturę modyfikowano, a próby zwiększenia wydajności mikroprocesora pociągały zwiększenie złożoności całego komputera. Obecnie mikroprocesor często determinuje architekturę całego komputera, a także system operacyjny dla niego.

Szybki rozwój technologii pozwala na podwojenie lub potrojenie w ciągu roku mocy obliczeniowej komputerów, bez wielkiego zwiększenia kosztów. W pracach nad tworzeniem coraz lepszych mikroprocesorów można zaobserwować kilka głównych kierunków. Wydajność procesora zależy głównie od stopnia scalenia, architektury wewnętrznej i sposobu połączenia go w danym systemie komputerowym.

OD TRANZYSTORA DO PROCESORA

Stopień scalenia jest niewątpliwą barierą na drodze do osiągnięcia powodzenia. Obecnie niektóre procesory (np. Intel 80386) zawierają ponad 275 000 tranzystorów przy szerokości ścieżek 1,5 μm. Teoretyczne granice możliwości technicznych nie są jeszcze osiągnięte – zastosowanie promieni X pozwoli prawdopodobnie uzyskać układy scalone o szerokości ścieżek ok. 0,5 μm.

Pewne problemy powstają przy próbie uzyskania systemów, w których szybkość funkcjonowania jednostki centralnej z bardzo szybkim zegarem byłaby wykorzystywana właściwie, tzn. aby reakcja pozostałych składników systemu była równie szybka.

Projektowanie i produkcja szybkich mikroprocesorów o wielkim stopniu scalenia wymaga doskonałej znajomości fizycznych właściwości użytych elementów, stosowania złożonych metod optymalizacji projektowania oraz zabezpieczenia przed możliwością powstawania błędów. Projektantowi mogą w tym pomóc złożone programy użytkowe CAD (projektowanie wspomagane komputerowo). Tego rodzaju programy działają głównie na stacjach roboczych lub na mikrokomputerach typu IBM PC/AT.

RISC CZY CISC?

Architektura wewnętrzna jest drugim czynnikiem wpływającym na wydajność procesora.

Od czasu, kiedy firmy IBM i Hewlett-Packard opracowały projekty swoich komputerów PC/RT i HP Precision, wiele firm komputerowych zaczęło interesować się nową koncepcją architektury komputerów – RISC. Polega ona nie tylko na ograniczeniu listy rozkazów, ale także na używaniu ich w taki sposób, aby każdy z nich mógł być wykonany w jednym cyklu maszynowym. Tę możliwość daje zastosowanie mechanizmu przetwarzania zakładkowego (ang. pipelining), który pozwala na jednoczesne wykonywanie kilku niezależnych rozkazów. Oczywiście, jest to możliwe tylko wówczas, gdy jednostka centralna nie jest zajęta tłumaczeniem mikrokodu, co ma miejsce w komputerach o architekturze CISC. Praktycznie, praca wykonywana w komputerach CISC przy interpretacji mikrokodu została w maszynach RISC jedynie przesunięta na czas kompilacji programu, ale właśnie to umożliwia uproszczenie procesora, a więc zwiększenie jego wydajności. W efekcie, twórcy kompilatorów będą musieli bardzo dobrze poznać dany procesor i sposób, w jaki traktuje on dane, aby stworzyć kompilator, który będzie z nim efektywnie współpracował.

Jednym z mikroprocesorów tworzonych według koncepcji RISC jest Clipper firmy Fairchild. Jest on wykonany w technologii CMOS o szerokości ścieżek 2 μm i zawiera 846 000 tranzystorów. Pracuje z częstotliwością zegara 33 MHZ. Osiąga szybkość przetwarzania 5–6Mips, a przy optymalnym użytkowaniu 33 Mips. Jest to układ o architekturze całkowicie 32-bitowej, ma oddzielne (niemultipleksowane) szyny danych i adresów. Zawiera 16 rejestrów użytkowych 32-bitowych, 16 rejestrów systemowych 32-bitowych oraz 8 rejestrów 64-bitowych przeznaczonych do operacji zmiennoprzecinkowych. Ponadto, Clipper ma dwa bufory pamięci podręcznej: 4 KB dla danych i 4 KB dla rozkazów, z możliwością pobierania rozkazów z wyprzedzeniem.

Ciekawostką tego procesora jest umieszczenie jednostki arytmetycznej dla liczb zmiennoprzecinkowych równolegle z jednostkami arytmetyczno-logicznymi dla liczb całkowitych tworzącymi trzy poziomy układu przetwarzania potokowego. Pozwala to wykonywać w ALU równocześnie cztery rozkazy. Tak zastosowany mechanizm przetwarzania potokowego powoduje, że Clipper może być porównywany pod względem wydajności z VAX 8600.

Zgodnie z założeniami koncepcji RISC Clipper ma 101 rozkazów, każdy wykonywany w jednym cyklu maszynowym. Oprócz tego, ma jeszcze dostępnych 67 makrorozkazów zapisanych w pamięci ROM procesora.

Ze względu na swoją dużą moc obliczeniową Clipper jest przeznaczony do pracy jako koprocesor, np. w zastosowaniach graficznych. Jego magistrala zewnętrzna może być zgodna ze standardami głównych istniejących procesorów. Obecnie projektowane są karty z Clipperem do IBM PC/AT oraz do stacji roboczej firmy Intergraph, przeznaczonej do prac CAD.

Idea firmy Fairchild w wypadku układu Clipper jest prosta: stworzenie procesora, który mógłby wzbogacić mikrokomputery typu IBM PC/AT oraz stacje robocze bez konieczności zmiany ich architektury. Clipper jest przeznaczony głównie dla języków wysokiego poziomu (Pascal, C) i pracy w systemie Unix.

Innym ciekawym przykładem procesora RISC jest Novix NC 4000 (por. Informatyka, nr 9, 1987). W tym wypadku celem twórców było zaprojektowanie procesora „rozumiejącego" język wysokiego poziomu – Forth. Naukowcy z firmy Novix wykorzystali w swoim projekcie zarówno możliwości, jakie dał wybrany przez nich język programowania, jak i mechanizmy stosowane przez inne firmy (przetwarzanie zakładkowe, przetwarzanie równoległe itp.).

Novix NC 4000 jest procesorem 16-bitowym, wykonanym w technologii HCMOS

Wyniki tekstów porównawczych komputera NC 4000 – czasy wykonania

Rodzaj testu	Motorola 68000, 8 MHz – asembler (1) i Forth (2)	Intel 80286, 10 MHz – język C	NC 4000 6 MHz – Forth
Milion pustych pętli	7 s	5 s	2,4 s (DO...LOOP) 0,17 s (FOR...NEXT)
Rekurencyjny ciąg Fibonacciego	19 s	1,2 s	0,19 s
Przełączenie zadań (praca wielozadaniowa)	–	17,5 μs	4,5 μs

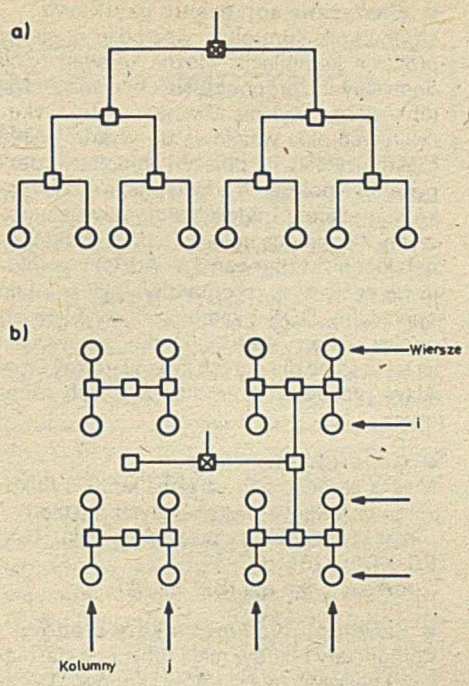
o szerokości ścieżek 3 μm. Zawiera 40 000 tranzystorów i pracuje z częstotliwością 6 MHz, choć ma się też ukazać wersja 8 MHz. Osiąga szybkość przetwarzania 10 Mips. Przygotowywana jest wersja 32-bitowa tego procesora, pracująca z częstotliwością 10 MHz, o szybkości przetwarzania 40 Mips.

NC 4000 ma możliwość adresowania pamięci 64 K × 16 bitów lub 4 MB przy wykorzystaniu portów rozszerzenia. Zawiera 15 rejestrów 16-bitowych, w tym jeden przeznaczony do obliczania pierwiastka kwadratowego.

Projekt firmy Novix doczekał się już praktycznej realizacji w formie karty dołączanej do PDP 11/73, w celu realizacji systemów ekspertowych w językach Forth i Pascal. Wyniki pierwszych testów przedstawiono w tabeli.

PRZETWARZANIE RÓWNOLEGŁE

Jednym z czynników decydujących o wydajności procesora jest sposób połączenia go w systemie. Obecnie wiele ośrodków badawczych zajmuje się tworzeniem systemów wieloprocesorowych przystosowanych do przetwarzania równoległego. Jedna z firm angielskich opracowała projekt systemu komputerowego do przetwarzania obrazów, opartego na równoległym przetwarzaniu informacji. W Paryżu opracowano nową metodę rozwiązywania równań z mechaniki płynów Navier'a-Stokes'a również opartą na przetwarzaniu równoległym, wykorzystując prostą komunikację między procesorami.



Połączenie procesorów przedstawione w postaci drzewa binarnego (a) i drzewa w postaci litery H (b)

Jednym ze sposobów połączenia procesorów w systemie wieloprocesorowym jest utworzenie drzewa binarnego. Systemy o architekturze drzewa są badane m. in. w Massachussets Institute of Technology. Takie systemy mogą w prosty sposób wykonywać operacje typu macierzowego. Wystarczy drzewo procesorów wyobrazić sobie w postaci litery H (rys.). Do zaadresowania każdego procesora p_{ij} wystarczy słowo o długości k (gdzie k jest liczbą poziomów drzewa) opisując ścieżkę dostępu do tego procesora.

. . .

Superprocesory takie, jak Clipper i NC 4000, są przeznaczone głównie do pracy w stacjach roboczych, typu Sun lub Xerox. Te zastosowania wymagają właśnie dużej mocy obliczeniowej, możliwości graficznych i dobrego systemu operacyjnego.

Prawdopodobnie za jakiś czas pojawią się procesory lepsze i szybsze od dzisiejszych. Nic bowiem nie wskazuje na to, aby pęd w kierunku tworzenia coraz lepszego sprzętu komputerowego miał zostać w najbliższym czasie zahamowany. Czy oznacza to jednak, że z wyborem procesora do konkretnego zastosowania mamy czekać, aż pojawią się te najlepsze?

Mikrokomputery biurowe i osobiste nie wymagają aż tak wyspecjalizowanych procesorów. W tym zastosowaniu sprawdziły się procesory „tradycyjne", o lepszym współczynniku wydajności do ceny. Z inżynierskiego punktu widzenia szuka się takiego rozwiązania danego problemu, które – spełniając nałożone warunki – będzie rozwiązaniem najprostszym i najtańszym.

Opracowała **DOROTA INKIELMAN**
na podstawie Micro-Systems, 1986–1987

Kompilator Ady dla IBM PC/AT

W 1980 r. Jean Ichbiach, główny projektant języka Ada, założył firmę Alsys. Pracuje w niej wielu jego współpracowników z grupy projektowej Ady. Powstał tam system kompilacji Ady na komputery osobiste rodziny IBM PC.

Budowa systemu

Kompilator Alsys Ada dla PC/AT pozwala programom aplikacyjnym pracować w trybie wirtualnym i uzyskiwać dostęp do rozszerzonej pamięci (16 MB) komputera PC/AT. Znaczy to, że duże programy mieszczą się na mikrokomputerze osobistym, a programiści nie muszą zajmować się dzieleniem ich na nakładki; znikają straty czasowe wynikłe z przemieszczania kodu nakładkowego do i z pamięci.

Na całość tego bardzo elastycznego systemu składają się następujące części:

- **Kompilator**
Jest to program, dla którego komputerem macierzystym jest IBM PC/AT. Tłumaczy każdą jednostkę kompilacji na kod wynikowy mikroprocesorów rodziny 8086.

- **Zarządca biblioteki**
Jest programem usługowym, służącym do tworzenia, usuwania, kopiowania i przemianowywania biblioteki programów Ady, stanowiącej skorowidz systemu operacyjnego DOS, w którym kompilator przechowuje dane o kompilacji.

- **Zarządca jednostek**
Stanowi program usługowy, sprawdzający i pielęgnujący pojedyncze jednostki w bibliotece programów Ady.

- **Program wiążący (ang. binder)**
Jest to program narzędziowy, sprawdzający i łączący oddzielnie kompilowane jednostki w pojedynczy program wynikowy, w formacie wymaganym przez PC/XT lub PC/AT. Generuje również wydruk wyników wiązania, obejmujących m. in. granice i wielkości jednostek, kolejność opracowania i błędy. Wiązanie jednostek obejmuje także konsolidację wykonaną przez konsolidator DOS, który łączy program wynikowy i system wykonawczy RTE w plik wykonywalny.

- **Egzekutor RTE (ang. run-time executive)**
Jest zbiorem procedur przydziału pamięci do przechowywania danych podczas wykonania programu w Adzie, obsługi wyjątków i sterowania zadaniami.

- **Karta z pamięcią**
Jest to karta z pamięcią 4 MB, dołączana przy sprzedaży systemu.

Właściwości kompilatora

- **Tworzenie dużych programów aplikacyjnych**
Kompilator Alsys Ada dla PC/AT został napisany w języku Ada, szczególnie nadającym się do pisania dużych programów. Umożli-

wia pracę programów aplikacyjnych w trybie wirtualnym, tj. umieszczenie w rozszerzonej pamięci zarówno danych jak i kodu. Ograniczenie przez system operacyjny wielkości programu do 640 KB nie istnieje i na komputerze klasy PC można pracować z dużymi programami bez dzielenia ich na nakładki.

● **Wytwarzanie efektywnego kodu**
Efektywność jest zawsze sprawą najważniejszą w projekcie kompilatora Ady. Celem końcowym jest wytwarzanie efektywnego, wysokiej jakości kodu wrodzonego. Podczas analizy postaci źródłowej programu Ady kompilator wykonuje kilka przebiegów, podczas których wykrywa błędy i zamienia instrukcje wysokiego poziomu na ich dwójkowe odpowiedniki – bardziej zwarte, pozbawione błędów, o strukturze dogodnej dla szybkiego wykonania. W odróżnieniu od pseudokodu wytwarzanego przez niektóre inne kompilatory, kod skompilowany przez system Alsys Ada jest bezpośrednio wykonywany przez mikroprocesor z pełną szybkością. Program aplikacyjny może pracować na komputerze docelowym w postaci modułu ładowalnego, bez instalowania oddzielnego systemu wspomagającego wykonanie.

● **Działanie egzekutora RTE**
RTE zwiększa efektywność kodu wynikowego podczas wykonywania programu. Na przykład, bardzo ekonomicznie przydziela i zwalnia pamięć na przechowywanie danych. RTE służy również do sterowania wielozadaniowością.

● **Elastyczne sprzężanie użytkowe**
Użytkownik kontroluje wszystkie opcje w procesie kompilacji. Można używać opcji domyślnych, proponowanych przez Alsys, lub wskazać własne. Można również wykorzystywać pliki wsadowe do wykonywania często używanych poleceń kompilacji. Bezpośrednie polecenia udostępniają środowisko kompilatora i wiele programów usługowych. Za pomocą poleceń o prostej, skróconej składni (wzorowanej na Adzie) wywołuje się kompilator, program wiążący, system operacyjny DOS i samouczek, wybiera się opcje i wykorzystuje biblioteki. Te polecenia są takie same dla innych kompilatorów firmy Alsys pracujących na stanowiskach roboczych.

● **Generowanie kodu**
Można zażądać wytwarzania kodu wynikowego w jednej z następujących postaci:
– rozkazy 8086 odpowiednie dla IBM PC/XT lub IBM PC/AT
– rozkazy tylko dla IBM PC/AT.

● **Szczegółowe komunikaty błędów**
Komunikaty błędów mają funkcje wyjaśniające i nie zawierają skrótów i kodów. Wyjaśnienia są napisane prostym językiem angielskim i pojawiają się na wydruku postaci źródłowej programu, a miejsce wystąpienia błędu jest dokładnie identyfikowane strzałką. Każdy komunikat jest omówiony w odpowiednim rozdziale podręcznika. Na życzenie, kompilator wyświetla szczegółowe, możliwe interpretacje zaistniałego problemu.

Dokumentacja
Do każdego kompilatora firma dołącza kompletną dokumentację obejmującą: instrukcję instalowania, podręcznik użytkowania, podręcznik Ady i dodatek zawierający opis cech zależnych od implementacji.

Instrukcja instalowania informuje, w jaki sposób przeprowadza się ładowanie i ustawianie parametrów kompilatora. Opisany jest również sposób postępowania przy dołączeniu dodatkowej pamięci do komputera IBM PC/AT.

Podręcznik użytkowania objaśnia proces kompilacji. Zawarte są w nim również wyjaśnienia dotyczące składni poleceń kompilowania modułów w środowisku systemu DOS, wiązania skompilowanych jednostek i konsolidacji powiązanego programu wynikowego z egzekutorem RTE na IBM PC/AT. Dodatkowo wyjaśniono używanie zarządców biblioteki i jednostek.

Podręcznik języka zawiera normę ANSI Ady (ANSI/MIL-STD 1815 A). Dodatek F podręcznika zawiera omówienie właściwości języka zależnych od implementacji i sposobów posługiwania się pragmą INTERFACE.

Serwis i konsultacje są prowadzone przez biura firmy znajdujące się we Francji, Wielkiej Brytanii i USA.

Opracował MARIUSZ KUC
na podstawie materiałów firmy Alsys

Rodzina superkomputerów ETA10

Amerykańska firma ETA Systems oferuje aktualnie cztery superkomputery ETA10. Superkomputer ETA10 może składać się maksymalnie z ośmiu procesorów centralnych, z których każdy jest wyposażony w pamięć centralną o pojemności 32 MB. Wszystkie procesory centralne mają dostęp do pamięci wspólnej (ang. shared memory) o pojemności od 64 do 2000 MB. Szybkość przesyłania danych między procesorem centralnym a pamięcią wspólną wynosi 9,1 Gbitów/s.

Cztery modele ETA10 różnią się czasem cyklu procesora centralnego. Czas cyklu najmniejszego modelu **P** wynosi 24 ns, modelu **Q** – 19 ns, modelu **E** – 10,5 ns i największego modelu **G** – 7 ns. Procesory centralne rodziny ETA10 wykonano w technologii CMOS zapewniającej niski pobór mocy. Każdy procesor centralny składa sięz 240 mikroukładów (łącznie prawie trzy miliony bramek logicznych) zamontowanych na jednej 44-warstwowej płycie. Modele **P** i **Q** są chłodzone powietrzem, modele **E** i **G** – ciekłym azotem. Najmniejszy superkomputer ETA10-P (jeden procesor) wykonuje maksymalnie 375 Mflops/s (milionów operacji zmiennoprzecinkowych na sekundę) i kosztuje milion dolarów, a największy, ETA10-G, wykonuje 10 286 Mflops/s i kosztuje dziewięć milionów. Dla porównania, moc obliczeniowa jednego procesora centralnego ETA10 odpowiada mocy pełnego systemu Cyber 205, firmy CDC, wyposażonego w dwa potoki wektorowe i pamięć o pojemności 4 megasłów. Rozmiary procesora centralnego ETA10 są jednak istotnie mniejsze i wynoszą 71×216×168 cm.

Superkomputery ETA10 można włączyć do sieci Ethernet, LCN lub Hyperchannel. Połączenie z sieciami Ethernet (10 Mbitów/s) i Hyperchannel (50 Mbitów/s) jest oparte na powszechnie stosowanym standardzie TCP/IP, natomiast połączenie z siecią LCN (50 Mbitów/s) wykorzystuje standard RHF (ang. remote host facility) firmy CDC.

Aktualny system operacyjny EOS jest zbliżony do systemu VSOS komputera Cyber 205. Nowy system operacyjny ETA System V będzie zgodny z Unixem i zostanie wyposażony w kompilator języka C. Oferowany kompilator Fortranu zapewnia automatyczną wektoryzację i uwzględnia zarówno rozszerzenia notacji macierzowej nowego standardu Fortranu 8x, jak i rozszerzenia Fortranu komputerów Cyber 205.

Oficjalne dane firmy ETA Systems mówią dotychczas o siedmiu sprzedanych systemach dla następujących instytucji: NASA Ames Laboratory, Uniwersytet Stanowy na Florydzie, Instytut Meteorologii w RFN, Centrum im. Johna von Neumanna, Centrum Superkomputerowe w Minnesocie, Ośrodek Obliczeniowy Uniwersytetu w Groningne (Holandia) oraz Tokijski Instytut Technologii. Dalsze zamówienia oraz wynik współzawodnictwa z superkomputerowym potentatem, firmą Cray Research, będą zależeć przede wszystkim od jakości zapowiadanego przez firmę oprogramowania podstawowego. Uważa się również, że cena i parametry najmniejszego superkomputera ETA10-P stanowią wyzwanie rzucone przez firmę ETA Systems producentom superkomputerów Alliant, Convex i SCS.

MM
na podstawie: Supercomputer, listopad 1987

Bardzo szybka jednostka centralna firmy Inmos

Brytyjska firma Inmos Corp. z Bristolu opracowała nową wersję swego słynnego transputera – podzespołu umożliwiającego składanie systemów o ogromnych możliwościach obliczeniowych. Nowy transputer o nazwie T800 zawiera 32-bitowy mikroprocesor T414 i tworzy jednostkę zmiennoprzecinkową o szybkości 4 milionów rozkazów na sekundę, jakiej nigdy dotąd nie udało się osiągnąć w pojedynczej kostce. Jest on zrealizowany w technologii CMOS, o rozstępie ścieżek 1,4 μm. Zawiera układ mnożący i mechanizm przesunięcia oraz trójmagistralową architekturę pozwalającą układowi zmiennoprzecinkowemu działać jednocześnie z jednostką centralną.

Jest to 4–12 razy szybciej aniżeli współczesne kombinacje procesora 32-bitowego i koprocesora zmiennoprzecinkowego w oddzielnych kostkach. Aby osiągnąć taką szybkość w innym rozwiązaniu, należałoby zbudować wielokostkowy układ obejmujący dodatkowy układ mnożący i pamięć, co oczywiście wprowadziłoby opóźnienia międzykostkowe. T800 ma wbudowaną statyczną pamięć operacyjną o pojemności 4 Kb. Układ Intel 80386 i koprocesor 80287 tworzą system 12 razy wolniejszy, kombinacja 68020/68881 Motoroli jest 6 razy wolniejsza, a połączenie kostek National Semiconductor 32032 i 32100 – 4 razy. Jedynie następna generacja układów Motoroli 68030 i 68882 jest tylko 2 razy wolniejsza od T800. Zespół o porównywalnej szybkości musiałby obejmować czterokostkowy układ standardowych 32-bitowych jednostek centralnych, specjalną kostkę sprzęgającą i dwukostkowy zestaw zmiennoprzecinkowy firmy Weitek o symbolach 1164 i 1165.

Układ T800 został opracowany w ramach europejskiego projektu architektury równoległych komputerów ESPRIT. Jest on częścią proponowanego superkomputera złożonego z transputerów. Już obecnie inna firma z Bristolu, Meiko Ltd., założona przez byłych pracowników Inmos, oferuje superkomputer o nazwie Computing Surface, który dotychczas wykorzystywał transputery T414, osiągając wówczas (przy 311 transputerach) szybkość 3 miliardy rozkazów na sekundę, a obecnie w tej samej konfiguracji, ale z wykorzystaniem T800, osiągnie 360 milionów operacji zmiennoprzecinkowych na sekundę. Te możliwości zmiennoprzecinkowe wydają się być podstawową zaletą nowej kostki, gdyż większość zastosowań jednostek 32-bitowych ma wymagania w tym zakresie.

Mikroukład, który miał się pojawić w sprzedaży na początku 1987 roku ma 84 końcówki i zawiera 300 tysięcy tranzystorów. Skuteczna długość kanału wynosi zaledwie 0,8 μm. Powierzchnia układu jest tylko o 25% większa od T414. Na obszarze o wymiarach 0,28×0,23 mm² projektanci zdołali umieścić nie tylko jednostkę centralną T414, ale także procesor zmiennoprzecinkowy i układy logiczne mnożenia i dzielenia. Stare, wielokostkowe rozwiązania układów zmiennoprzecinkowych powstawały, gdy technologia była na poziomie małego i średniego stopnia scalenia. Analiza wykazała, że pewne tradycyjne bloki są mało wydajne lub zajmują zbyt wiele miejsca. Uwzględniając specyfikę systemu zmiennoprzecinkowego (więcej dostępów do argumentów aniżeli działań) uzyskano nowe podejście projektowe, co dało w rezultacie optymalne rozwiązanie, charakteryzujące się przede wszystkim ścisłym sprzężeniem między zespołem zmiennoprzecinkowym i jednostką centralną.

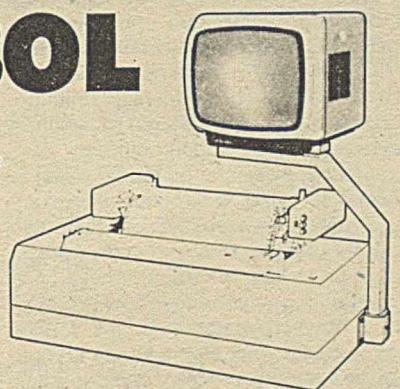
Oparty na T800 pakiet dla IBM PC i systemów kompatybilnych pozwala na realizację typowych, zmiennoprzecinkowych programów wzorcowych (ang. benchmark) w czasie 2–3 sekund, natomiast przy podobnym

dokończenie na s. 30

ZEKOM proponuje użytkownikom drukarek DZM-180

MONITORY EKRANOWE

MV 1664
MV 2580L



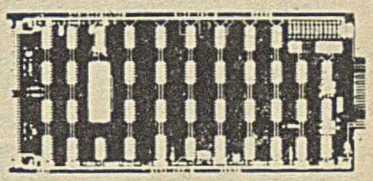
Nasze rozwiązania zostały sprawdzone we współpracy z wieloma systemami
– między innymi jako konsole operatorskie ODRA-1300, R-32, MERA-400.

Stosując nasze monitory:
* zaoszczędzisz papier,
* zmniejszysz hałas,
* uodpornisz system na awarię drukarki,
* i co najważniejsze – zwiększysz efektywność swojej pracy!

ZEKOM
ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ
Skr. pocztowa 35, 90-955 Łódź 8, tel. 57-25-83

ZEKOM poleca użytkownikom komputerów MERA-400

PAKIET SPRZĘGU
SV 24/400
/SM-UZDAT-11/



PAKIET SPRZĘGU SV24/400 służy do przyłączenia dodatkowego terminala do jednostki centralnej komputera jako odpowiednik pakietu UZDAT-11.
Pakiet obsługuje przerwania operatorskie.

ZEKOM

EO/1088/88

Zagrożenie planów opracowania szybkich komputerów w Japonii

Ogłoszone w 1981 r. plany opracowania bardzo szybkich komputerów w Japonii, które wówczas wywołały popłoch wśród zachodnich rządów i wytwórców komputerów, zostały ostatnio ograniczone na skutek trudności technicznych i cięć budżetowych.

Gdy ogłoszono „Superspeed Project" o wartości 140 mln dolarów, zapowiedziano uzyskanie superkomputera o szybkości 100 do 1000 razy większej od istniejących wówczas modeli. Pod koniec 1986 r., kiedy pozostały niecałe trzy lata z ośmiu przeznaczonych na realizację projektu, okazało się, że zamierzenia te należało znacznie ograniczyć, a założenia projektowe prototypowego komputera nie zostały jeszcze ustalone.

Nowy dyrektor tego projektu, Jira Shibata, powiedział co prawda, że „nie jest rozczarowany" i że harmonogram prac jest w zasadzie przestrzegany, ale rozmowy z dwudziestoma uczestnikami projektu ujawniły, że zrezygnowano z jednej z trzech technologii, w jakich opracowywane były nowe elementy, a dwie pozostałe opracowuje się znacznie wolniej i przy większych kosztach niż planowano. Dlatego projektanci obecnie przewidują, że w większości zostaną zastosowane konwencjonalne kostki krzemowe, a tylko niektóre podzespoły będą wytworzone w nowych technologiach.

Ponadto, opracowanie oprogramowania do tego komputera okazuje się zadaniem tak złożonym, że wielu informatyków japońskich uważa, iż jeżeli nawet prototyp zostanie opracowany, to trudno będzie go doprowadzić do stanu praktycznej używalności.

Trudności, jakie napotkał omawiany projekt, dotknęły również jeszcze ambitniejszego programu komputerów piątej generacji, którego celem jest opracowanie komputerów o działaniu zbliżonym do rozumu ludzkiego. Okazało się jednak, że prace prowadzone w zakresie tzw. sztucznej inteligencji natrafiły na poważne problemy.

Te wszystkie niepowodzenia skłaniają część ekspertów do wniosku, że byłoby korzystniej, gdyby wspomnianymi projektami zajęły się poszczególne firmy, a nie rząd, który na skutek bieżących trudności musi ograniczać nakłady.

Wyrażany jest też pogląd, że trudności z realizacją superkomputerów oznaczają zakończenie dynamicznego okresu rozwoju technicznego, kiedy wydawał się możliwy nieograniczony wzrost we wszystkich kierunkach. Jest mało prawdopodobne, by Japonia łatwo mogła powtórzyć taki sukces, jakim w latach siedemdziesiątych był projekt układów VLSI, który pozwolił japońskim producentom półprzewodników zapanować na rynku kostek pamięciowych.

Nie oznacza to oczywiście rezygnacji z omawianych projektów. Postępy w rozwoju superkomputerów pociągają za sobą rozwój całego przemysłu komputerowego i dlatego są one krytycznym odcinkiem całej mikroelektroniki, oprogramowania i systemów komputerowych. Poza tym rośnie zapotrzebowanie na superkomputery w wielu dziedzinach – od meteorologii do projektowania samochodów i astrofizyki. Japońskie firmy Hitachi, Fujitsu i NEC wytwarzają i sprzedają od kilku lat superkomputery, aczkolwiek nie stały się one przedmiotem znaczącego eksportu.

W Stanach Zjednoczonych największe instalacje superkomputerów należą do Departamentu Obrony, który stosuje je głównie do opracowywania broni jądrowych, oraz Krajowej Agencji Bezpieczeństwa – do utajniania i odtajniania informacji. Ogłoszenie projektów japońskich spowodowało przyspieszenie podobnych prac w USA, a ostatnio w ramach SDI (ang. Strategic Defense Initiative – inicjatywa obrony strategicznej) finansuje się wiele małych przedsiębiorstw pracujących w dziedzinach związanych z superkomputerami.

W Europie też są znane co najmniej dwa projekty wspomagane przez rządy, dotyczące technologii półprzewodników niezbędnych do osiągnięcia znacznego postępu w szybkości obliczeń.

Zakres badań nad superkomputerami jest w USA szerszy aniżeli w Japonii, zwłaszcza w dziedzinie przetwarzania równoległego. Japończycy koncentrują się nad opracowaniem szybkiego komputera uniwersalnego. Celem Superspeed Project jest prototyp uniwersalnej maszyny o szybkości obliczeniowej rzędu 10 miliardów operacji zmiennoprzecinkowych (ang. gigaflops) na sekundę. Najszybsze obecnie superkomputery osiągają od jednej dziesiątej do jednej piątej tej wartości i to tylko w krótkich odcinkach czasu. Zwiększenie szybkości obliczeniowej planuje się uzyskać zarówno przez opracowanie szybkich kostek komputerowych przy wykorzystaniu nowych sposobów projektowania i nowych materiałów, jak i przez zmiany w tradycyjnej architekturze komputera, gdzie centralny procesor wykonuje każdy rozkaz krok po kroku, a wynik przechowuje w pamięci przed pobraniem go do następnego obliczenia. Nowe podejście oznacza konieczność utworzenia zupełnie nowego oprogramowania, które powinno umożliwiać podzielenie problemu tak, aby mógł być podjęty przez wielokrotne procesory jednocześnie.

Uważa się, że stworzono już podstawy tego nowego systemu. Według wstępnych opracowań Superspeed Project, technologia kostek jest obecnie tylko udoskonalana, a architektura i oprogramowanie powinny być ukończone w ciągu roku. *„W roku 1987 koncepcja naszego superkomputera powinna być określona szczegółowo"* – powiedział p. Shibata. Naukowcy zaangażowani w tym projekcie nie w pełni podzielają optymizm szefa.

Jedną z nowych technologii kostek jest tzw. złącze Josephsona, wymagające obniżenia temperatury do – 170°C, przy której gwałtownie maleje oporność elektryczna niektórych materiałów. Trudności techniczne, jakie tu wystąpiły, były tak duże, że firma IBM już przed kilku laty zrezygnowała z opracowania tej technologii, a obecnie czynią to Japończycy.

Dwie pozostałe technologie to HEMT (ang. High Electron Mobility Transistors – tranzystory z szybkimi elektronami), która też wymaga obniżonych temperatur, i kostki z arsenku galu. I tu jednak projektanci nie osiągnęli takiego upakowania układów w kostce, by możliwe były założone szybkości.

Liczba procesorów w kompletnym projekcie jest z kolei ograniczona względami finansowymi. Oficjalnie na projekt przeznacza się około 140 mln dolarów, ale w rzeczywistości do dyspozycji jest tylko połowa tej kwoty. Według firmy Fujitsu mówi się o 16 podzespołach przetwarzających, podczas gdy wiadomo, że w podobnych urządzeniach w USA stosuje się od 64 do kilkuset procesorów.

Te wątpliwości techniczne i niepewności dotyczące oprogramowania stawiają pod znakiem zapytania spełnienie założeń na szybki japoński superkomputer.

Opracował JAN RYŻKO
Na podstawie
International Herald Tribune
z 20–30 listopada 1986 r.

dokończenie ze s. 29

pakiecie z wykorzystaniem transputera T414 wykonanie ich trwa 28–30 sekund, a przy zmiennoprzecinkowym koprocesorze Intel 80287 – 135 sekund.

Na czele kilkunastoosobowego zespołu opracowującego T800 stali: liczący 35 lat David May, odpowiedzialny za architekturę wyrobów Inmos i 32-letni Tony Fuge, który od 1984 roku kieruje projektowaniem transputerów. Natomiast głównym projektantem superkomputera Computing Surface w Meiko Ltd. jest zaledwie 25-letni Moray McLaren.

JR
na podstawie
Electronics, 27. 11. 1986

System plików (1)

Wydawało się, że przemianowanie terminu **zbiór** na plik już się powszechnie przyjęło. Okazało się jednak, że nie jest to takie oczywiste. Jeszcze w wielu ośrodkach starszy personel kopiuje, „dampuje" i „skreczuje" zbiory albo „fajle". Również dyskusja podjęta na łamach INFORMATYKI (nr 5, 1988) oraz Mikroklanu (nr 1, 1988) świadczy, że nic nie jest pewne.

Dlatego też, przy okazji postanowiłem szerzej przedstawić terminologię związaną z tym pojęciem. Trzeba tutaj bowiem dodać, że terminologia angielska była kształtowana przez różne firmy – oczywiście w najszerszym zakresie przez IBM – a w związku z tym często to samo pojęcie było różnie nazywane. Co więcej, na dobór terminów miał wpływ rozwój cech funkcjonalnych systemu plików, zapoczątkowany plikami będącymi sekwencją bloków na taśmie magnetycznej.

W książkach o systemach operacyjnych plik jest kolekcją danych. Proponuję jednak nawiązać do dziedziny programowania (ostatecznie system operacyjny jest programem) i przyjąć następującą definicję pliku:

Plik (file) jest abstrakcyjnym typem danych, którego struktura danych jest zlokalizowana na nośniku określonego urządzenia, a operacje są realizowane przez system operacyjny.

Wyróżniamy następujące poziomy zarządzania informacjami przechowywanymi w plikach. **System plików** (file system management) zarządza dostępem do informacji zawartych w pojedynczych plikach oraz ich rozmieszczeniem na nośniku urządzenia pojedynczego lub ich grupy. **System zarządzania danymi** (data management system) zajmuje się organizowaniem informacji zawartych w pojedynczych plikach w złożone struktury danych, umożliwiające ich wykorzystanie według różnorodnych metod dostępu. **System bazy danych** (database system) tworzy złożone struktury informacyjne, wiążące informacje zlokalizowane w wielu plikach. Moduły systemu plików oraz niekiedy systemu zarządzania danymi są z reguły autonomicznym programem implementującym jedną z trzech podstawowych organizacji bazy danych – **relacyjną** (relational), **hierarchiczną** (hierarchical) lub **sieciową** (network).

Dalej ograniczę się tylko do prezentacji podstawowych pojęć związanych z systemem plików.

Organizacja logiczna (logical organization) pliku określa sposób rozmieszczenia jednostek (units) informacji w pliku. Dla użytkownika pliku ze względu na interpretację jednostki informacji wyróżniamy pliki:
– **znakowe** (character), w których jednostką informacji jest znak, najczęściej o kodzie ASCII,
– **binarne, dwójkowe** (binary), w których jednostką informacji jest bajt o wartości z zakresu 0...255, będący najczęściej częścią danej zawartej w kolejnych bajtach, tworzącej słowo lub wartość liczbową,
– **zakodowane** (coded), w których jednostką informacji jest też bajt o wartości będącej kodem interpretowanym przez program wykorzystujący ten plik (jak np. tekst programu w BASICu składowany przez interpretator).
Wewnętrznie jednostką informacji jest więc bajt (8- lub czasami 7-bitowy).

Logicznie, jednostki informacji tworzą w pliku sekwencje zwane **rekordami** (record) o stałej lub zmiennej liczności jednostek. Wyróżniamy więc następujące organizacje logiczne:
– **sekwencyjne** (sequential) z uporządkowaniem rekordów według kolejności zapisu, z charakterystycznym oznacznikiem końca pliku (end of file),
– **strumieniowe** (stream) będące organizacjami sekwencyjnymi z rekordami zawierającymi pojedyncze jednostki informacji; organizacja ta jest charakterystyczna dla plików związanych z urządzeniami znakowymi (np. klawiaturą, ekranem itp.), ale nie tylko,
– **regionalne** (region), polegające na istnieniu określonej liczby ponumerowanych regionów, z których każdy może być pusty lub zajęty przez jeden lub więcej zapisanych rekordów (lub strumieniem); utożsamienie regionu z pojedynczym rekordem jest nazywane organizacją **względną** (relative) – w obu wypadkach plik jest fragmentami wypełniany informacją,
– **indeksowe** (indexed) lub **kluczowe** (keyed) będące zbiorem rekordów, z których każdy jest identyfikowany unikalnym kluczem (indeksem); istnieją też organizacje **indeksowo-sekwencyjne** (indexed-sequential), grupujące rekordy w sekwencje o tym samym podzakresie wartości kluczy.

Dodatkowym, ważnym atrybutem organizacji logicznej jest sposób dostępu do rekordów pliku. Wyróżniamy dwa takie sposoby:
– **sekwencyjny** (sequential), zapisujący lub odczytujący rekordy do/z pliku w kolejności, zawsze począwszy od początku pliku,
– **swobodny** (random), zapisujący lub odczytujący jednostki informacji w miejsce pliku, określone adresem logicznym (logical address); dla organizacji rekordowej adresem jest numer kolejny rekordu, dla regionu – numer regionu (w regionie mamy dostęp sekwencyjny).

Tutaj konieczne jest wyjaśnienie przyjęcia terminu **swobodny**, odpowiadającego angielskiemu random – losowy, bezpośredni.

Używanie pojęcia **losowy** nie ma większego sensu, gdyż nie ma tutaj żadnej przypadkowości, bowiem zapisujemy lub odczytujemy konkretnie wskazany rekord. Często używane pojęcie „bezpośredni" jest bardziej związane z możliwością zapisu lub odczytu wskazanego sektora (lub bloku logicznego) na dysku, niż z dostępem do określonego adresem logicznym rekordu czy regionu. Co więcej, implementacja systemu plików w systemie operacyjnym może udostępniać dostęp swobodny, czytając sekwencyjnie plik. Przy realizowaniu dostępu sekwencyjnego system plików wykorzystuje dostęp bezpośredni do sektora dysku, o wyliczonym przez niego adresie fizycznym. Stąd też należy rozróżnić pojęcie dostępu na poziomie logicznym, nazywając go **dostępem swobodnym** (random access) od dostępu na poziomie fizycznym, nazywając go **dostępem bezpośrednim** (direct access).

Teraz możemy już dla każdego systemu jednoznacznie opisać organizację pliku określoną przez niego. Przykładowo, system MS-DOS dostarcza typ plikowy o organizacji strumieniowej, z dostępem sekwencyjnym, z jednostkami informacji będącymi bajtami, a więc dostępne są pliki znakowe, binarne i zakodowane. Interesujące jest, że np. translator BASICu, działający pod tym systemem, dostarcza dwa typy plikowe o organizacji logicznej:
– **strumieniowej** z dostępem sekwencyjnym i znakowym typem informacji,
– **rekordowej** z dostępem sekwencyjnym oraz swobodnym i binarnym typem informacji.

W następnej części zostaną omówione terminy związane z implementacją systemu plików.

WACŁAW ISZKOWSKI

Terminologia ta została opracowana na podstawie następujących pozycji:

[1] Iszkowski W., Kalinowska-Iszkowska M., Maniecki M.: Projektowanie systemów operacyjnych w ujęciu syntetycznym. PWN 1987
[2] Madnick S. E., Donoven J. J.: Systemy operacyjne. PWN 1983
[3] Show A. C.: Projektowanie logiczne systemów operacyjnych. WNT 1980.

Zabrodzki J.: Cyfrowe generowanie obrazów (1)

INFORMATYKA 1988, nr 6, s. 1

Pierwsza część charakterystyki rozwiązań konstrukcyjnych cyfrowego generowania obrazów z wykorzystaniem techniki rastrowej.

Zabrodzki J: Digital image generation (1)

INFORMATYKA 1988, No. 6, p. 1

First part of characteristics of digital image generation hardware solutions applying raster technology.

Zabrodzki J.: Digitale Bildgenerierung (1)

INFORMATYKA 1988, Nr. 6, S. 1

Erster Teil einer Charakteristik von Hardwarelösungen der digitalen Bildgenerierung mit Anwendung der Rastertechnik.

Griswold R. E., Griswold M. T.: Języki wysokiego poziomu do przetwarzania napisów – COMIT, SNOBOL4 i Icon (3)

INFORMATYKA 1988, nr 6, s. 4

Trzecia część charakterystyki metod i narzędzi do przetwarzania napisów, zawierająca omówienie języka Icon.

Griswold R. E., Griswold M. T: High level languages for string processing – COMIT, SNOBOL4 and Icon (3)

INFORMATYKA 1988, No. 6, p. 4

Third part of characteristics of methods and tools for string processing, which contains discussion of the Icon language.

Griswold R. E., Griswold M. T.: Höhere Sprachen zur Zeichenfolgenverarbeitung – COMIT, SNOBOL4 und Icon (3)

INFORMATYKA 1988, Nr. 6, S. 4

Dritter Teil einer Charakteristik von Methoden und Hilfsmittel zur Zeichenfolgenverarbeitung, der eine Besprechung der Icon-Sprache umfasst.

Turowicz A.: Lokalna sieć komputerowa D-Link (1)

INFORMATYKA 1988, nr 6, s. 7

Pierwsza część charakterystyki sieci D-Link, obejmująca omówienie jej parametrów technicznych oraz oprogramowania.

Turowicz A.: D-Link local area network (1)

INFORMATYKA 1988, No. 6, p. 7

First part of the D-Link network characteristics, which contains discussion of its technical parameters and software.

Turowicz A.: D-Link-Lokalnetz (1)

INFORMATYKA 1988, Nr. 6, S. 7

Erster Teil einer Charakteristik von dem D-Link-Computernetz, der eine Besprechung seiner technischen Parameter und Software umfasst.

Caban D.: Transputer – narzędzie przetwarzania równoległego

INFORMATYKA 1988, nr 6, s. 10

Szczegółowe omówienie nowatorskich rozwiązań organizacji przetwarzania, architektury i programowania układu transputera.

Caban D.: Transputer – tool for parallel processing

INFORMATYKA 1988, No. 6, p. 10

Detailed discussion of innovatory solutions of transputer circuit's processing organization, architecture and programming.

Caban D.: Transputer – ein Hilfsmittel für parallele Verarbeitung

INFORMATYKA 1988, Nr. 6, S. 10

Detaillierte Besprechung von bahnbrechenden Lösungen der Transputerschaltung im Bereich ihrer Verarbeitungsorganization, Architektur und Programmierung.

Huzar A., Kaliś A.: Sieć komputerowa Jednolitego Systemu SKJS2

INFORMATYKA 1988, nr 6, s. 14

Omówienie rozwiązań sprzętowych i programowych sieci komputerów Jednolitego Systemu, zrealizowanej przez Centrum Obliczeniowe Politechniki Wrocławskiej.

Huzar A., Kaliś A.: SKJS2 Unified System network

INFORMATYKA 1988, No. 6, p. 14

Discussion of hardware and software solutions of the network for Unified System computers, which is realized by Computing Center of the Wrocław Technical University.

Huzar A., Kaliś.: SKJS2 – ein ESER-Computernetz

INFORMATYKA 1988, Nr. 6, S. 14

Eine Besprechung von Hard- und Softwarelösungen der ersten Version von dem SKJS2-ESER-Computernetz, die von Rechenzentrum der Technischen Universität in Wrocław realisiert wurde.

Ciechomska H., Wójciekian T., Zaborowska E.: PROMIS – System Zarządzania Bazą Danych (2)

INFORMATYKA 1988, nr 6, s. 16

Druga część charakterystyki systemu zarządzania bazą danych dla minikomputerów klasy SM-4, zawierająca omówienie pozostałych ważniejszych składników tego systemu.

Ciechomska H., Wójciekian T., Zaborowska E.: PROMIS – a data base management system (2)

INFORMATYKA 1988, No. 6, p. 16

Second part of characteristics of the data base management system for SM-4 minicomputer, which contains discussion of more important remaining components of the system.

Ciechomska H., Wójciekian T., Zaborowska E.: PROMIS – ein Datenbankverwaltungssystem (2)

INFORMATYKA 1988, Nr. 6, S. 16

Zweiter Teil einer Charakteristik von Datenbankverwaltungssystem für SM-4-Kleinrecher, der eine Besprechung von wichtigeren restlichen Bestandteilen des Systems umfasst.

Bielecki J.: Turbo C. Preprocesor i makrodefinicje

INFORMATYKA 1988, nr 6, s. 19

Sposoby i przykłady użycia preprocesora oraz makrodefinicji w języku Turbo C.

Bielecki J.: Turbo C. Preprocessor and macrodefinitions

INFORMATYKA 1988, No. 6, p. 19

Methods and examples for preprocessor and macrodefinitions application in the Turbo C language.

Bielecki J.: Turbo C. Preprozessor und Makrodefinitionen

INFORMATYKA 1988, Nr. 6, S. 19

Methoden und Beispiele der Preprozessor- und Makrodefinitionenanwendung in der Turbo C-Sprache.

6-letnie doświadczenie software'owe sprawdzone w ponad 1000 zakładach pracy

TYLKO **SYSTEM CSK/32**

jeśli zarządzanie Twoim przedsiębiorstwem wymaga minikomputera o mocy obliczeniowej Odry oraz kilkunastu skomputeryzowanych stanowisk pracy

System CSK/32 to:

MIKROKOMPUTERY QUATRO CSK/32
— przystosowane do pracy z kilkunastoma terminalami, pracujące w dowolnym trybie graficznym

OPROGRAMOWANIE
— systemowe WDOS, MIKROLAN
— narzędziowe — MEGA BLOK, TABPLAN, PL-TEKST, BGRAF, TRYS
— aplikacyjne — FK K & K, EM K & K, KADRY, PŁACE

WDROŻENIA, SZKOLENIA
— zespoły ds. wdrożeń i szkoleń pracują w:
Gdyni tel. 248 018, tlx 054792 ● Krakowie tel. 373 573 ● Poznaniu tel. 676 271
Wrocławiu tel. 481 679 ● Warszawie tel. 258 528, tlx 816711
Sopocie — w wyspecjalizowanym Salonie Sprzedaży, ul. Kombatantów 1

CSK™

computer studio kajkowscy

EO/74/88

7

MIĘDZYWOJEWÓDZKA SPÓŁDZIELNIA PRACY „SIÓDEMKA"

ŁÓDŹ, AL. KOŚCIUSZKI 93,

ZAKŁAD INFORMATYKI I SYSTEMÓW KOMPUTEROWYCH

poleca po **najniższych cenach** w kraju:

— mikrokomputery 8-, 16-, 32-bitowe najwyższej jakości renomowanych firm z całego świata
— urządzenia peryferyjne:
- drukarki — również 24-igłowe i laserowe
- streamery
- napędy dyskowe 3", 5.25"
- monitory monochromatyczne, kolorowe, EGA, HEGA, VGA
- karty rozszerzenia pamięci
- kontrolery
- dyski twarde typu Winchester 20 MB, 40 MB, 60 MB, 80 MB

— systemy wielodostępne i lokalne sieci mikrokomputerowe:
- MULTI-LINK
- D-LINK
- XENIX

— materiały eksploatacyjne:
- dyskietki 5.25" MD2-D
- dyskietki 5.25" MD2-HD
- dyskietki 3" CF2
- dyskietki 3.5" MF 2DD
- taśmy barwiące do wszystkich typów drukarek STAR i NEC

Termin realizacji zamówień **natychmiast** po złożeniu zamówienia. **Bezpłatnie:** szkolenia, kursy, zestawy oprogramowania narzędziowego i użytkowego — przy dostarczeniu kompletnych systemów.

Proponujemy również na wszelkiego rodzaju mikrokomputery programy wspomagające zarządzanie przedsiębiorstwem:

— system finansowo-księgowo-kosztowy
— system zbytu i zaopatrzenia
— system technicznego przygotowania produkcji
— system materiałowy
— system kadrowy i kadrowo-płacowy (również dla pracowników akordowych)
Wymienione systemy pracują w wersjach sieciowych i wielodostępnych.

Wszelkich informacji udzielamy codziennie (oprócz sobót) w siedzibie Zakładu w Łodzi przy Al. Kościuszki 101, tel. 36-51-00, w godzinach 8—16.

EO/1005/87

SPÓŁKA z o. o.

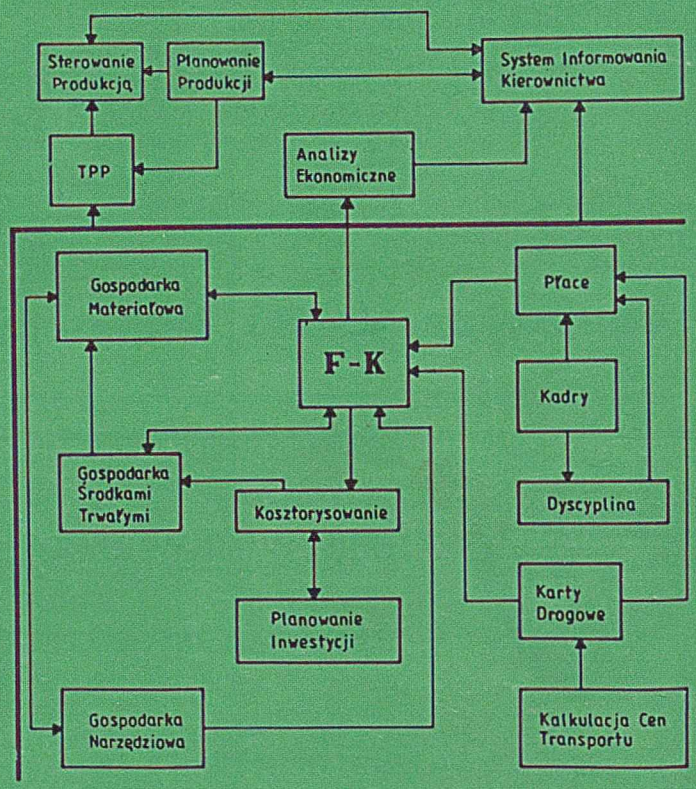
PRZEDSIĘBIORSTWO
INNOWACYJNO
WDROŻENIOWE

INTERAMS

00–867 Warszawa, ul. Chłodna 35/37
tel.: 24 78 16, 24 78 23, teleks: 81 56 24

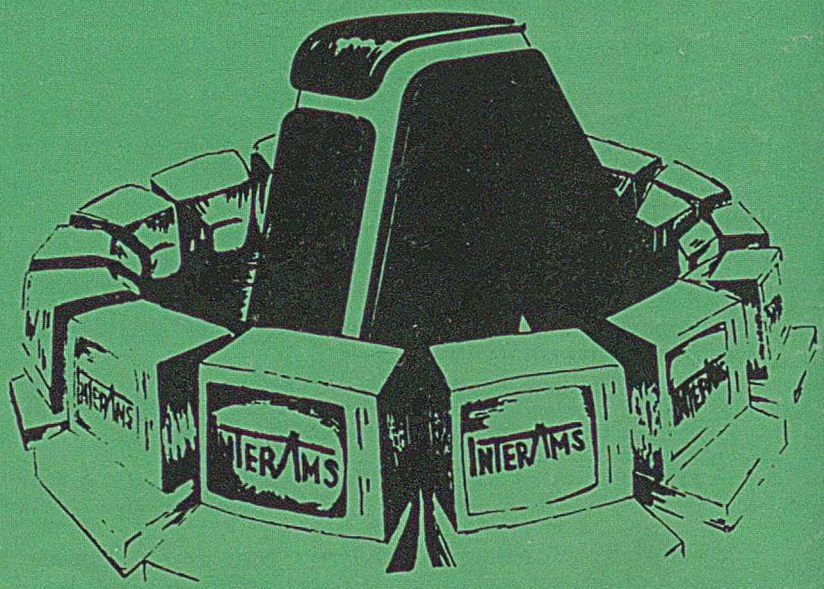
OFERUJE:

- *Zintegrowany pakiet systemów do zarządzania przedsiębiorstwem*
- *Usługi w zakresie kompleksowej komputeryzacji przedsiębiorstw*
- *Sprawdzone rozwiązania sieciowe np.: sieć „Novell"*
- *Kursy dla menadżerów*
- *Bank oprogramowania*
- *Sprzęt komputerowy najwyższej jakości z gwarancją do 1,5 roku*



ZAPRASZAMY...

do naszego salonu na prezentację sprzętu i oprogramowania



EO/390/88